Unidad 2: SISTEMAS OPERATIVOS.
Funciones. Características. Clasificación. Componentes. Arquitecturas.

Sistemas Operativos

Capas de un Sistema Operativo

Hemos visto que un **SO** es un **conjunto de programas y rutinas** que permiten **controlar las operaciones** de la computadora con un **mínimo de intervención humana**.

Todos los componentes del SO se ordenan en una **estructura de capas o niveles** para definir claramente las interconexiones entre ellos.

El SO tiene programas o rutinas residentes en memoria (Kernel o Núcleo, que se carga al encender la máquina, con el Bootstrapping o I.P.L.), el resto se encuentra en el disco del sistema.

Al **Núcleo** del sistema también se lo llama Residente, Monitor, Supervisor, Kernel, etc., debido a que es **uno de los programas principales** que componen el SO.

Normalmente está **compuesto por módulos o rutinas que controlan los recursos físicos y lógicos** de un computador (Procesadores, Memoria Central y Secundaria, Datos, Archivos, Dispositivos Periféricos, etc.).

Los compiladores, cargadores, debuggers y otros módulos del Software de base (utilitarios) usualmente no se consideran parte de los SO al igual que los sistemas de Administración de Bases de Datos y el software de control de comunicaciones.

En cambio el sistema de acceso de comunicaciones y los programas de servicios son fuertemente dependientes del SO y de sus interfases.

El SO generalmente se divide en **dos niveles de administradores**:

1. PRIMER NIVEL (Administrador de Pedidos)

Los programas y rutinas de este nivel son los responsables de la **administración de los pedidos** y normalmente se los engloba en un solo módulo.

1.1 Funciones Administrador de Pedidos (JOB SCHEDULER)

Este módulo, al que se lo conoce como **Job Scheduller o Planificador de Trabajos**, es el responsable de:

- Cargar programas.
- Crear procesos.
- Administrar y controlar los accesos de los usuarios.
- Establecer las protecciones.

Es el módulo que **decide a qué trabajo asignarle** el estado de **" listo para la ejecución"**.

En algunos sistemas, este módulo se complementa con un lenguaje (**Job Control Language**) que **permite programar las actividades del computador**.

2. SEGUNDO NIVEL (Administrador de Recursos)

Este nivel se conforma por **cuatro o cinco módulos** de acuerdo a los distintos tipos de recursos que debe administrar.

2.1. Administración del procesador

Consta básicamente de **dos módulos**:

2.1.1. Planificador de procesadores y procesos (Dispatcher)

Decide, una vez llegado un trabajo dividido en procesos, **a qué procesador asignar el proceso** que tiene que ser ejecutado y le otorga el uso del procesador.

2.1.2. Controlador de tráfico (Traffic Controler)

Este se encarga de **crear, modificar y actualizar el contexto asociado a un proceso** para posibilitar su pasaje entre los diferentes estados. Por ejemplo, al iniciarse un proceso, previamente pasa por el **controlador de tráfico**, el cual genera las respectivas actualizaciones, la asignación de memoria y recursos para su ejecución.

2.2. Administración de memoria

Se encarga de **asignar a los procesos la memoria necesaria** para su ejecución.

2.3. Administración de periféricos

Comprenden todos los módulos necesarios para la utilización de los periféricos. Ejemplo: asignación de dispositivos, spooler, controladores, etc.

2.4. Administración de información (*File System*)

Se refiere a las rutinas que permite **manipular y manejar el sistema de archivos**, permite accederlos (leerlos, grabarlos, crearlos, destruirlos, copiarlos, renombrarlos, etc.).

2.5. Administración de comunicaciones (*Communication Manager*)

Es el responsable de **compartir los recursos distribuidos** mediante una **red de computadoras**.

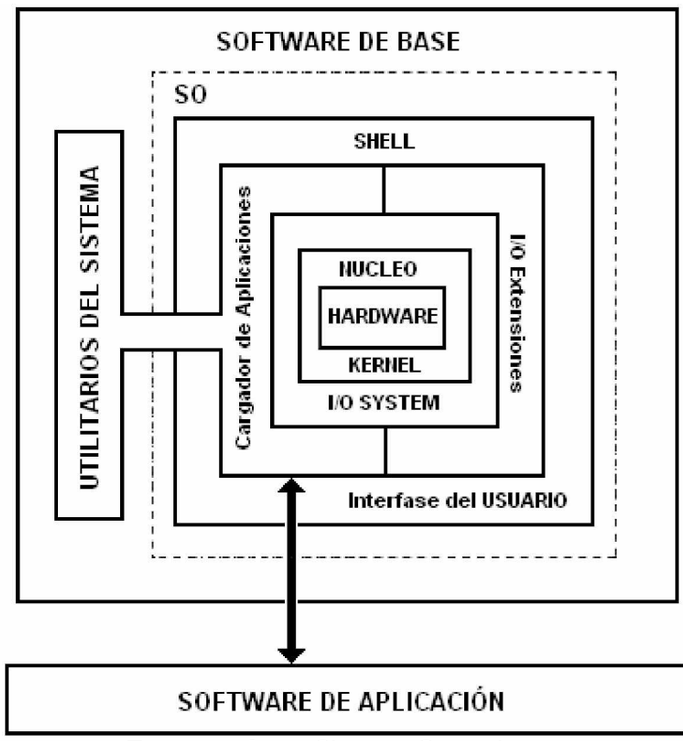
FUNCIONES DE LOS ADMINISTRADORES DE RECURSOS

- Mantener actualizado un registro o contabilidad del estado o uso de los recursos.
- Cumplir (en función de la política implementada a tal fin) con las demandas, decidiendo quién, cómo, cuándo y por cuánto tiempo recibe determinado recurso.
- Asignar dicho recurso a quien lo demande.
- Recuperar el recurso después que se ha utilizado.

3. TERCER NIVEL

Este nivel aparece en **equipos grandes y algunos medianos**, **multiprogramados**, y se ocupa de:

- Las transacciones del Kernel.
- El Sistema de Acceso de Comunicaciones
- El Sistema de Multiprocesamiento.



Esquema de un Sistema Operativo

Requerimientos de un SO

- Simple
- Portable.
- Estructurado (modular).
- Confiable.
- Soporte de múltiples usuarios o procesos.

- Soporte de red o procesamiento distribuido.

Funciones de un Sistema Operativo.

Después de todo lo visto, podemos decir que los objetivos de los SO son básicamente tres

1. Inicialización
2. Máquina extendida
3. Administración de recursos brindando servicios

1. Inicialización

Inicializar la máquina

Esta tarea es **llevada a cabo por el Kernel**, con **rutinas residentes** en memoria **ROM** o **RAM** de la máquina **más otras residentes en el disco** del sistema.

La inicialización tiene por **objetivo preparar la máquina real** y llevarla a un estado tal que **pueda ejecutar el primer trabajo**.

TIPOS DE INICIALIZACIÓN

1.1. Inicialización Total (cold start)

Este proceso, al que también se lo conoce como **Bootstrapping** o **Inicial Program Loader (IPL)**, se ejecuta al encender la máquina.

Primero pone una dirección determinada en el Registro de Direcciones (Program Counter PC) del Procesador Central que busca en la memoria ROM un código perteneciente a la rutina de inicialización del Kernel.

Se carga luego el resto del SO, que permanecerá residente en memoria RAM, completándose el IPL ("***Booteo***" en la jerga informática) y así la máquina queda preparada para ejecutar la primer carga de trabajo bajo el control del SO que luego administrará todo el procesamiento del sistema de cómputo hasta su desconexión.

Aproximadamente el 1% de las tareas del Sistema está dedicada a esta función.

PASOS DEL PROCESO

- **Verificación de recursos**

  Consiste en la creación de las tablas de recursos disponibles en el sistema (discos, impresoras, terminales, etc.). Esta verificación la realiza el Sistema Operativo para luego poder administrar dichos recursos cuando son requeridos por los programas a ejecutar.

- **IPL (Initial Program Loader o Booteo).**

  Luego de la verificación de recursos, se carga desde el disco el resto del SO (que queda residente en la memoria RAM completándose así el proceso de IPL).

  La máquina queda preparada para su uso presentando en la pantalla el símbolo (***Prompt***) respectivo y aguarda el ingreso de un comando por el dispositivo de entrada (teclado, mouse, etc.).

1.2. Inicialización parcial (Warm Start)

En este caso no crea ni inicializa las tablas generadas por el proceso de Verificación de recursos en el instante del "booteo".

La recuperación depende generalmente del tipo o clase de evento ocurrido que provocó la interrupción de la ejecución y de ello depende qué trabajo preservado reinicia.

Los eventos causantes de la interrupción de la ejecución pueden ser generados por errores del Hardware, violaciones a Protecciones, corte de luz, finalización del tiempo de uso de CPU, etc.

El esquema para la inicialización parcial es importante en los casos de Sistemas en Tiempo Real, en grandes Sistemas Multiusuarios, o en Arquitecturas Tolerantes a fallas (fault-tolerance systems). En este último caso depende de la operatividad de los equipos.

Proceso de carga del SO en una PC

Cuando se enciende el ordenador, se ejecuta un conjunto programas o rutinas (***rutinas del BIOS*** o Basic Input Output System) que están grabados en la memoria de sólo lectura (ROM, EPROM o EEPROM). Por la característica física en que están grabadas estas rutinas (por hardware y no por software) se las denomina ***firmware***.

Otra característica que destaca a las rutinas del BIOS del resto de los programas, es que las primeras no necesitan de un sistema operativo para funcionar. Esta cualidad les da el nombre de ***stand alone***.

Una de las rutinas del BIOS, el ***POST*** (Power On Self Test) tiene como función realizar un autodiagnóstico de encendido

Durante la POST, la computadora identifica su memoria, sus discos, su teclado, su sistema de vídeo y cualquier otro dispositivo conectado a ella. Si por algún motivo, la computadora no es operable el BIOS ejecuta otra rutina denominada ***SetUp*** (Configurador) que se encarga de mantener y actualizar una ***tabla de configuración*** que se encuentra almacenada en una memoria pequeña y volátil denominada ***CMOS***, y que mantiene su carga gracias a una pila que le da energía mientras la computadora está apagada.

La rutina SetUp, también puede ser disparada en forma voluntaria por el usuario utilizando una combinación de teclas.

Si el POST no encuentra dificultades, se ejecuta otra rutina del BIOS llamada ***IPL*** (Initial Program Loade – Cargador Inicial de Programas).

La función del IPL es buscar en el ***MBR*** (Master Boot Record – Registro Principal de Arranque) la ***Tabla de Particiones***, y en ella información acerca de la ***partición activa***.

Con esta información puede encontrar al ***Booter*** que es una rutina que no pertenece al BIOS sino que es el cargador del sistema operativo.

Las características del Booter son:

- Es software y no firmware.
- Es stand alone.

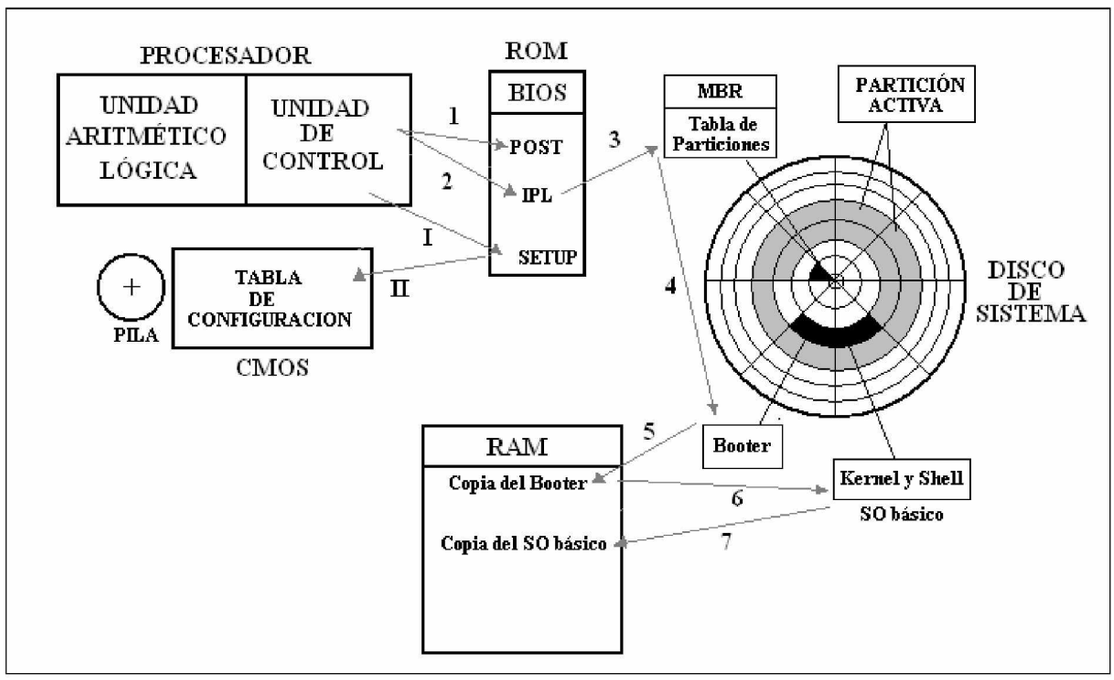
Cada sistema operativo tiene un Booter propio.

Encontrado el Booter, se carga una copia de este en memoria RAM y se ejecuta.

Cuando el Booter entra en acción busca a los archivos que conforman el sistema operativo básico (kernel y shell), realiza una copia de estos en memoria RAM y los ejecuta.

Una vez que el sistema operativo básico entra en funcionamiento, toma el control del sistema y termina el proceso de arranque.

Al proceso de carga del sistema operativo de denomina ***BOOTING*** y al disco que lo contiene se lo denomina "***SYSTEM DISK***" o "***BOOT DISK***".



Proceso de arranque de una PC

REFERENCIAS

La UC ordena que se ejecute el POST Se ejecuta el Monitor o IPL El IPL busca los datos de la partición activa en la tabla de particiones que se encuentra en el MBR Con los datos obtenidos, el IPL busca el Booter en le comienzo de la partición activa	Se copia el Booter en la RAM El Booter busca al sistema operativo básico (kernel y shell) en la partición activa Se hace una copia del sistema operativo básico en la RAM, toma el control da la máquina.	La rutina SETUP sólo se dispara si el POST encuentra problemas durante el examen de los dispositivos o cuando el Usuario lo desea. El SETUP actualiza los datos que se encuentran en la tabla de configuración alojada en la CMOS.

CARACTERÍSTICAS DE LAS RUTINAS DEL BIOS	**CARACTERÍSTICAS DEL BOOTER**
• Firmware • Stand Alone	• Software • Stand Alone

2. Máquina extendida o Interfase hombre-máquina.

La máquina extendida es un módulo del SO llamado **Shell** o **Intérprete de Comandos**, que **actúa como medio de comunicación entre el usuario y el hardware**, transformando a la máquina real en una máquina más fácil de manejar, es decir, más amigable (**User Friendly)**

Entre el 7 al 9 % de las tareas del Sistema está dedicada a esta función.

2.1. Funciones principales de la máquina extendida

- **Separar la complejidad de la máquina desnuda del usuario**. Una parte del SO se ocupa de separar la complejidad del Hardware y lo transforma en una **máquina virtual** más fácil de usar con una interfase para el usuario más amigable y menos compleja.

- **Actuar como interfase de Entrada/Salida (E/S)** y controlar el complejo manejo de los dispositivos de E/S cuando el usuario requiere de un servicio.
- **Facilitar la comunicación con el usuario**, ya sea a través de la **Shell** o del **Job Control Language (JCL)**, **de iconos, intérprete de Comandos u Ordenes**.
- Aceptar entradas de nuevos Trabajos.

La **Shell** es la parte visible del SO y puede ser intercambiable.

Las interfases con la máquina dependen del hardware. Las instrucciones del SO (**Systems Calls**) se ejecutan a través de **primitivas** (instrucciones pertenecientes a un lenguaje de programación que se ejecutan completas, sin ser interrumpidas o divididas).

2.2. Visiones del SO por parte de Shell

2.2.1. Visión del Usuario Común (User)

El SO provee la interfase que proporciona una visión global y abstracta del computador ocultando su complejo funcionamiento interno.

Provee una función muy importante, que es, proteger el uso de los recursos entre él o los usuarios y el sistema para garantizar la integridad y el adecuado reparto de los mismos.

El SO es un conjunto de comandos u órdenes que ofrecen una gran variedad de servicios para que el usuario pueda ejecutar sus tareas en el sistema de cómputo (por ejemplo copiar o renombrar archivos, ejecutar diversos programas, etc.).

La interacción del usuario con el sistema de cómputo siempre se realiza a través de la **Shell**, la cual transmite al SO los requisitos del usuario al sistema (en algunos sistemas se lo llama **Administrador de Trabajos** o **Job Scheduler**).

2.2.2. Visión del Administrador del Sistema de Cómputo (System Manager o Super-User)

Un Sistema Operativo es por excelencia el administrador de todos los recursos ofrecidos por el hardware para alcanzar un eficaz rendimiento de los mismos en cuanto a su uso.

Los Recursos fundamentales son: Procesadores, Memorias, Entrada - Salida, Información, Comunicaciones, etc.

El **System Manager** configura al SO para que el Sistema funcione eficientemente y los usuarios puedan compartir los recursos. Es el responsable de que el Sistema de cómputo funcione con el adecuado rendimiento y con seguridad.

El **System Manager** también suele llamarse **Superusuario** (**Superuser**).

Habría dos casos más (**Programador del Software de Base** y **Diseñador del SO**) que son poco frecuentes y pueden verse como usuarios especiales que desarrollan su programación como una aplicación particular.

2.3. TIPOS DE SHELL

2.3.1. CLI (Command Line Interface)

Es una interfase basada en **comandos** u **órdenes llamada** que permite ingresar un conjunto de caracteres que el SO interpretará como comandos u órdenes.

Ejemplo: Shell del D.O.S.

2.3.2. GUI (Graphical User Interface)

Permite accesos a través de iconos o gráficos.

Ejemplo: Escritorio de Windows.

3. Administrador de Recursos

Más del 90% del SO se dedica a esta función.

3.1. Objetivos

- Facilitar a los usuarios compartir y proteger recursos.
- Optimizar el porcentaje de utilización de los recursos.

Para ello el SO controla todos los objetos de un sistema de cómputo complejo (usuarios, procesadores. memoria, discos, etc.) en cuanto a quién usa cuál recurso y su respectiva planificación del reparto. De esta forma va asignando y desasignando el uso de estos recursos e intercede como ***arbitro imparcial*** en los conflictos que se generan entre los programas y usuarios por el uso compartido.

Se dice que el SO implementa

1. Una **Política** dado que asigna prioridades (de uso y/o de acceso a los recursos).
2. Una **Estrategia** ya que ordena los accesos y los conflictos.
3. Una **Autoridad** pues debe recuperar los recursos otorgados a los procesos y ordenar el uso de los mismos.
4. Una **Protección** brindando seguridad a los usuarios entre sí y preservando la integridad de los recursos.
5. Una **Contabilidad** para llevar el control del uso y disponibilidad de los recursos.

Y el SO ofrece

a) **Facilidades para crear, manipular y eliminar objetos** sobre los que se quiere realizar operaciones, a través de la Gestión y Conservación de la Información sobre ellos.

b) **Un ambiente para la ejecución de trabajos**, mediante la gestión del conjunto de recursos que permiten ejecutar los mismos.

c) **Facilidades para compartir el conjunto de recursos entre los usuarios**, mediante un planeamiento y ordenamiento de los trabajos.

CARACTERÍSTICAS COMUNES DE LOS SO

1. GESTIÓN Y REPARTO DEL CONJUNTO DE RECURSOS

El término recurso se refiere a Tiempo de CPU, E/S, reloj, Memoria Central, Datos, Variables, Archivos, etc.

1.1. CLASIFICACIÓN DE LOS RECURSOS SEGÚN SU NATURALEZA

- **Físicos**: CPU, Memoria.
- **Lógicos**: Tiempos, Variables.

1.2. CLASIFICACIÓN DE LOS RECURSOS SEGÚN SU USO

- **No Compartibles**: Uso restringido a un solo Proceso (Impresora, Unidad de Cinta, etc.).
- **Compartibles:** Empleados por varios procesos en forma concurrente (CPU, Archivos de Lectura-escritura, etc.).

1.3. OBJETIVOS DE LAS POLÍTICAS DE ASIGNACIÓN DE RECURSOS

- Mejor uso del Hardware.
- Satisfacer a los usuarios (Optimizando el tiempo de respuesta y servicios, etc.)

2. GESTIÓN DE LA INFORMACIÓN

Los objetivos de la Gestión de la Información son los siguientes:

2.1. **ENLACE (Binding)**

Asocia el objeto a los **espacios físicos** que puede direccionar un Procesador (para consultarlo, modificarlo, operarlo, etc.) sin ambigüedades.

2.2. ASIGNACIÓN DE DESCRIPTORES (DELAY BINDING TIME)

Permite retrasar la elección de un recurso.

2.3. MANEJO DE DIRECCIONES (VIRTUALES o REALES)

Ejemplo: Traslación estática o Dinámica (en Segmentación – Paginación).

3. COOPERACIÓN ENTRE LOS PROCESOS.

Los procesos interactúan generando conflictos en los siguientes casos:

- Compiten por el uso de los Recursos.
- Cooperan para alcanzar el objetivo de ejecutar las tareas del User.
- Comparten objetos.
- Se Comunican

Estos problemas, generalmente, se plantean en los accesos a los recursos escasos en donde se visualizan los conflictos por lo que se solucionan a través de herramientas de sincronización y de comunicaciones ofrecidos por el SO

4. PROTECCIÓN

Objetivo: Garantizar la integridad de los recursos y de los procesos.

Para resolver este conflicto el SO ofrece los siguientes mecanismos:

- **Ejecución dual de instrucciones:** Maestro - Esclavo.
- **Mutua Exclusión:** consiste en asegurar que los recursos compartidos sean accedidos por un solo proceso a la vez bajo ciertas condiciones.

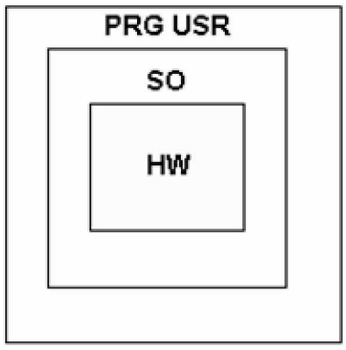
Arquitectura (Estructura) de un SO.

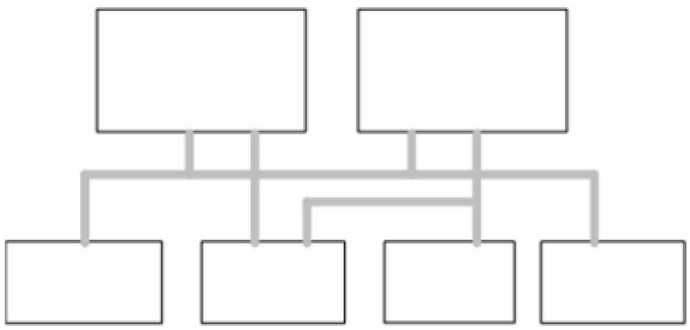
Generalmente cuando se mencionan las arquitecturas o estructuras de los sistemas operativos se refieren a la forma en que se organizan los distintos módulos en estratos o niveles.

Básicamente los SO tienen dos interfases: una con el Hardware (llamada Kernel o Núcleo) y otra con los usuarios y sus programas (llamada Shell).

1. Estructura tradicional o monolítica

Es la estructura de los **primeros SO** constituidos fundamentalmente por **un solo programa** compuesto de un conjunto de rutinas entrelazadas de tal forma que **cada una puede llamar a cualquier otra**.





1.1. Características

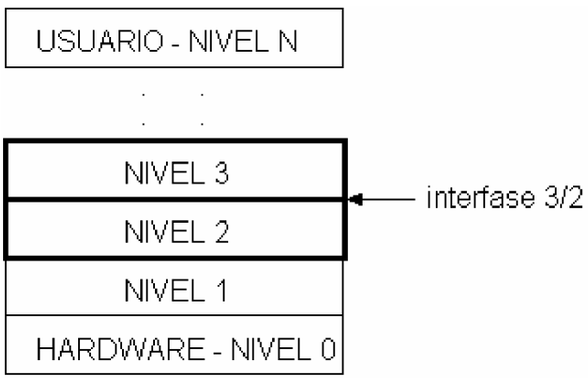
- Interfaces y niveles de funcionalidad no están bien separadas. No tienen estructura interna definida. Ejemplo: Aplicaciones pueden acceder a rutinas básicas de E/S para escribir directamente en la pantalla y unidades de disco.
- El SO es un conjunto de rutinas que se llaman entre sí libremente.
- No existe modularización, ni ocultamiento de la información.
- Difícil la reconfiguración y actualización.
- Carecen de protecciones y privilegios.
- Existe una mínima estructura. Servicios del sistema se piden a través de llamadas especiales (**supervisor call** o **System cal**).

El Kernel proporciona la planificación de la CPU y otras funciones del SO mediante las llamadas al sistema que utilizan los programas del usuario cuando se ejecutan.

Los programas del sistema, que hacen de interfase con el usuario, utilizan las llamadas al sistema proporcionadas por el Kernel para ofrecer funciones útiles como la compilación, etc.

Ejemplos: MS-DOS, Primeras versiones de UNIX®

2. Estructura en Estratos o Jerárquica



2.1. Características

- El principio de ordenamiento jerárquico.
- Los distintos módulos se organizan en una jerarquía de niveles.

- El módulo de nivel N funciona usando provistos los servicios por el nivel N-1.
- Cada módulo no sabe ni necesita saber como se implementan los servicios del módulo inferior, sólo conoce la interfase. Se oculta la existencia de ciertas estructuras de datos, operaciones y hardware a niveles superiores.
- SO mantiene un mayor control sobre computadoras y aplicaciones que lo utilizan.
- Se usa un enfoque descendente, así, de esta manera se determinan funciones y características globales y se las separa en componentes.
- El sistema es dividido en módulos, utilizando un "enfoque por capas", que divide al SO en varias capas (niveles), cada una construida sobre las inferiores. Donde cada capa constituye una implantación de un objeto abstracto que contiene datos y operaciones que pueden manipular esos datos.

2.2. Ventajas

a) Facilita la protección.

Cada capa consiste en algunas estructuras de datos y un conjunto de rutinas que pueden ser invocadas por las capas de niveles superiores y a la vez esa capa invocar operaciones de capas inferiores (rutinas no se invocan libremente y sin orden como en la estructura monolítica).

b) Permite implementar el principio de ocultamiento de la información "information hiding"

Permite a programadores realizar rutinas de bajo nivel que consideren adecuadas, siempre que las interfases externas permanezcan sin cambios y la rutina lleve a cabo la tarea indicada.

c) Facilita la sustitución y verificación de componentes (modularización)

Las capas se seleccionan de manera que cada una utilice funciones (operaciones) y servicios de las capas inferiores. Así se facilita la depuración del sistema. La explicación es lógica: el primer nivel se depura sin preocuparse por el resto del sistema ya que solo utiliza el hardware. Se pasa al segundo nivel sabiendo que el primero está correcto. Si hubiera algún error se sabe que es del nivel que se está depurando pues los inferiores no presentaban errores.

2.3. Desventajas

a) La definición de los distintos niveles es difícil

Se debe realizar una planificación cuidadosa. Ejemplo: el manejador de dispositivo para memoria auxiliar debe ser información para las rutinas de administración de memoria, ya que éstas requieren capacidad de utilizar memoria auxiliar, etc.

b) Algunas comunicaciones entorpecidas por la jerarquía

Por ej.: si el CPU Scheduler puede requerir un acceso a disco y sin embargo CPU Scheduler suele estar en un nivel inferior al I/O Scheduler.

c) Son más lentos

Ejemplos: OS/2 descendiente directo del MS-DOS, el cual fue creado para superar las limitaciones de éste y agregar multitarea y operación de modo dual, etc. Al adicionar complejidad y tener un hardware mucho más potente se implantó al SO en forma de capas.

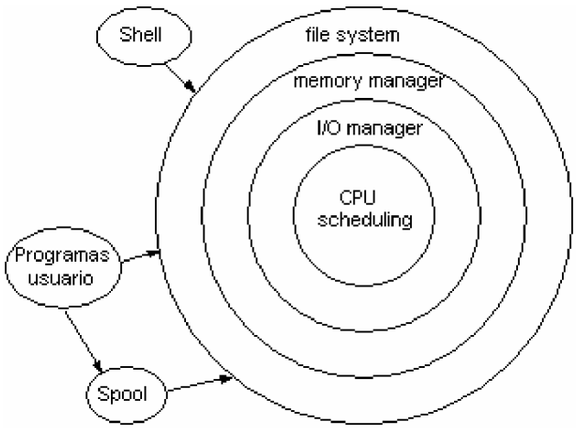
3. Anillos concéntricos (Rings)

Es otra **forma jerárquica muy similar a la anterior**.

3.1. Características

- Cada **ring** tiene una apertura (**trap**) por donde se accede desde las capas superiores a las inferiores.

- Las capas internas son más privilegiadas que las externas (protegidas).

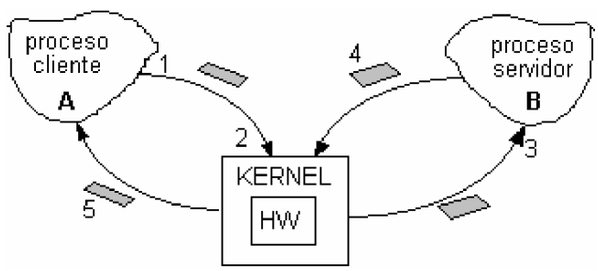


La solución actual de los SO de estructuras jerárquicas es diseñar menos capas pero con mayor funcionalidad, ofreciendo las mayorías de las ventajas del código modularizado a la vez que se evitan los difíciles problemas de la definición e interacción de las capas.

4. Estructura Cliente / Servidor

Se basa en lo mismo que el resto de los SO convencionales: El Kernel y los Procesos, presentando **grandes diferencias en cuanto a la forma de distribuir los trabajos entre sus distintas partes**.

- Se remueve la mayor cantidad posible de código del SO dejando un **Kernel mínimo** (*microkernel architectures*)
- **Funciones del SO** implementadas como **programas usuario**.
- Los servicios se implementan mediante la técnica de **message passing**.



La secuencia para este esquema es:

1) Proceso cliente (**A**) envía mensaje solicitando un servicio (por ejemplo: lectura de bloque en disco).
2) Kernel recibe el mensaje, toma las decisiones de planificación correspondientes y lo envía al proceso servidor (**B**).
3) Proceso servidor recibe el mensaje y ejecuta la función solicitada (en éste caso lee el bloque del disco).
4) Proceso servidor devuelve un mensaje al Kernel con el resultado de la operación
5) Nuevamente el Kernel recibe el mensaje de respuesta y lo envía al proceso cliente indicando que el servicio se ha cumplido con éxito o no.

Las funciones del Kernel son pocas. Entre ellas:

a) Tratamiento de interrupciones.
b) Multiprogramación.
c) Sincronización de mensajes.

Ventajas

a) **El sistema altamente modular:** los componentes son más chicos y manejables.

b) **Los módulos del sistema no tienen acceso directo al hardware:** un error en un servidor no tira el Sistema de cómputo abajo sino solo al servidor en cuestión.

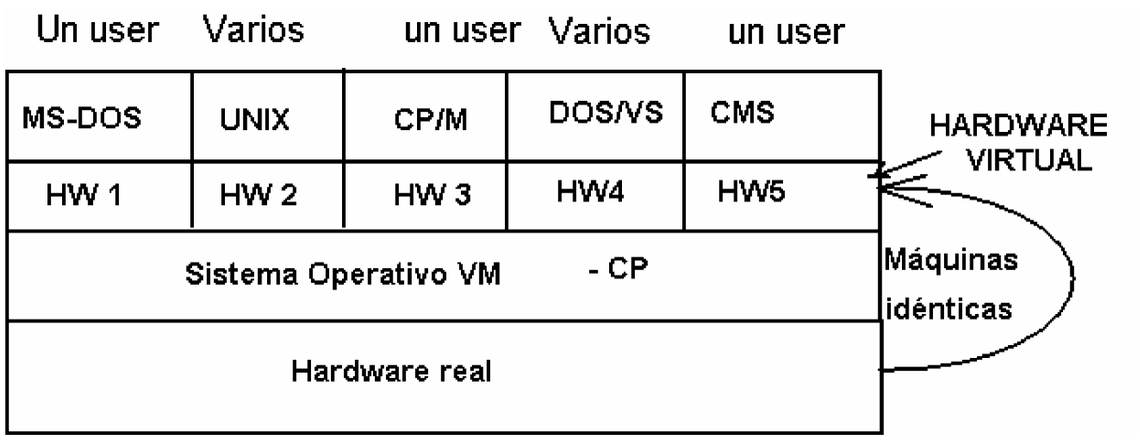
c) **Son especialmente útiles en ambientes distribuidos**.

Desventajas

a) **Algunos módulos del SO no pueden ser implementados como procesos usuario:** drivers de dispositivos, ciertas partes del administrador de memoria (Memory Manager).

Ejemplos: QNX® (diseñado en Canadá), Mach.

5. Máquinas virtuales



5.1. Características

- Basado en el mismo principio que las estructuras en estratos ya presentado.
- En lugar de proveer una visión simplificada del hardware, el SO crea *máquinas virtuales*: varias copias **idénticas** del hardware base.
- Cada una de estas máquinas virtuales tiene todas las características de la CPU real (memoria, interrupciones, puertos de E/S, etc.)
- A cada proceso se le otorga una copia (virtual) de la computadora subyacente.
- Por lo tanto, en cada máquina virtual se puede ejecutar cualquier SO que funcione en el hardware real.
- El núcleo de estos SO se denomina **monitor virtual**. Realiza la multiprogramación presentando tantas máquinas virtuales como se soliciten.
- Separan dos conceptos que suelen estar unidos en el resto de los sistemas:
  1. La multiprogramación.
  2. La máquina extendida.

5.2. Objetivo

- Integrar distintos SO dando la sensación de ser varias máquinas diferentes.

5.3. Ventajas

a) Cada usuario del sistema puede usar un SO distinto.
b) Permite un alto nivel de protección. Todas las máquinas virtuales son independientes.
c) Son especialmente útiles para diseño y desarrollo de Sistemas Operativos.

5.4 Desventajas

a) Difícil Implementación. Se necesita un duplicado exacto del hardware subyacente. Por ejemplo, como máquina física tiene dos modos, también debe poseerlos la máquina virtual.

b) La simulación del hardware real es muy costosa en recursos.

   Por ejemplo si una máquina física tiene tres manejadores de disco pero quiere dar servicio a siete máquinas virtuales; no puede asignar una unidad de disco a cada una, pero el software de la máquina virtual necesita un espacio considerable en disco para ofrecer memoria virtual y spoolers. En este caso se ha podido dar solución mediante **discos virtuales**, idénticos en todos los aspectos menos en el tamaño.

c) Cada máquina virtual es más lenta que la máquina real.

   La CPU se multiprograma entre varias máquinas virtuales, lo que las hace más lentas.

Ejemplo: El SO V.M. para IBM® es el mejor ejemplo de este concepto de máquinas virtuales.

Conclusiones

- **Ningún tipo de SO es mejor que otro**, cada uno presta un servicio especial y constituye un compromiso entre distintas alternativas.
- Para cubrir los distintos trabajos en un centro de cómputos se debería requerir distintos Sistemas Operativos, o uno solo que ofrezca una amplia gama de servicios.
- Los SO de propósitos generales son habitualmente más complejos y extensos que los especializados.