

Kernel

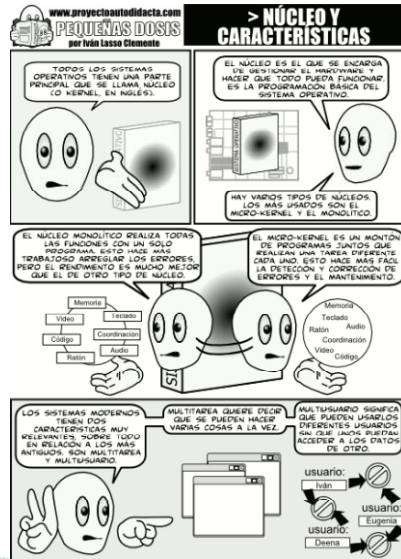
- ▶ Todos los sistemas operativos tienen una parte principal que se llama núcleo (o kernel, en inglés).
- ▶ El **núcleo** es el que se encarga de gestionar el hardware y hacer que todo pueda funcionar.
 - Es la programación básica del sistema operativo.



Estructura del núcleo del SO

- ▶ Conjunto de reglas normas y procedimientos
 - que especifican
 - las interrelaciones entre los componentes de un SI
 - y las características que deben cumplir cada uno de estos componentes
- ▶ Estructura Monolítica
- ▶ Estructura Jerárquica
- ▶ Estructura en Anillo
- ▶ Maquina Virtual
- ▶ Estructura Cliente-Servidor

El núcleo

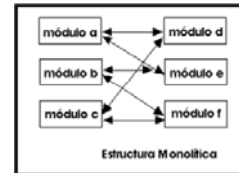


Estructura del núcleo

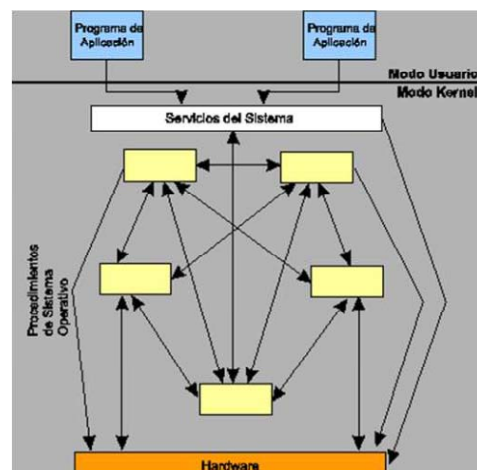
- ▶ Conjunto de reglas, normas y procedimientos
 - que especifican
 - las interrelaciones entre los componentes de un SI
 - y las características que deben cumplir cada uno de estos componentes
- ▶ Estructura Monolítica
- ▶ Estructura Jerárquica
- ▶ Estructura en Anillo
- ▶ Máquina Virtual
- ▶ Estructura Cliente-Servidor

Estructura Monolítica

- ▶ Es la estructura utilizada en los primeros SSOO
- ▶ Su estructura consiste en que no existe una estructura como tal.
- ▶ El SO está constituido por un único programa compuesto
 - de multitud de rutinas interrelacionadas entre sí
 - de forma que cada una pueda llamar a cualquier otra.
- ▶ El núcleo tiende a ser de gran tamaño
 - aumentando las posibilidades de fallo (caídas del sistema).
- ▶ Difícil configuración y actualización
 - Falta de protecciones y privilegios en rutinas de acceso a memoria, disco, etc..
- ▶ Buena definición de parámetros entre rutinas
- ▶ Hechos a medida
 - Eficientes y rápidos
 - Poco flexible



Ejemplo de SO Monolítico



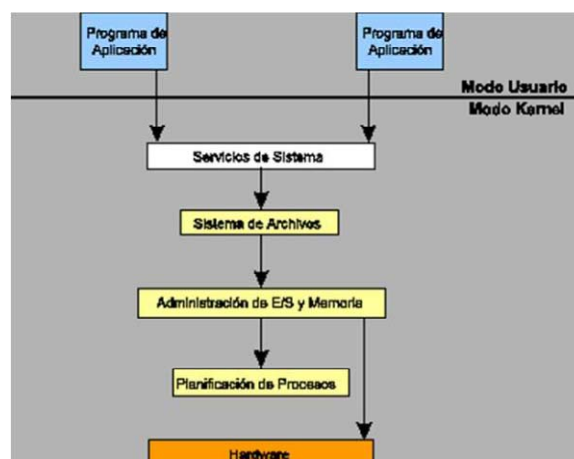
Estructura jerárquica

- ▶ A medida que los SSOO fueron creciendo, fue siendo necesaria una mayor estructuración.
- ▶ Es una estructura jerárquica
 - que se divide en distintos niveles
- ▶ Pequeños módulos
 - perfectamente definidos
 - Perfectamente clara la relación con otros módulos (interface)
- ▶ El módulo de cada nivel funciona utilizando los servicios del nivel inferior
 - Facilita la protección y el acceso al sistema
- ▶ Se basan la mayor parte de sistemas actuales

Capa 5 - Usuario
Capa 4 - Archivos
Capa 3 - Entrada/Salida
Capa 2 - Comunicaciones
Capa 1 - Memoria
Capa 0 - Gestión CPU
Capa 1 - Hardware

Sistema jerárquico THE

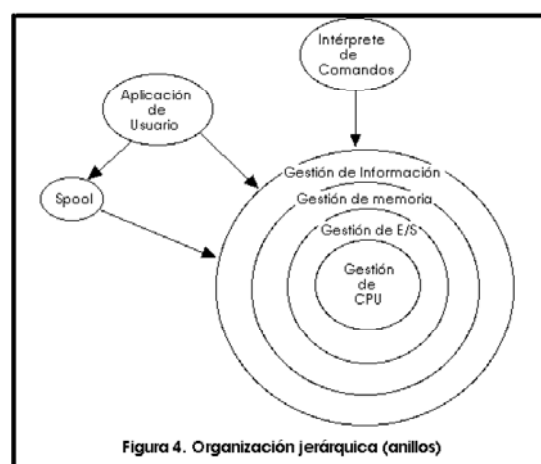
Ejemplo de SO Jerárquico



Estructura en anillos

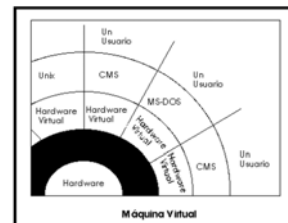
- ▶ Es una evolución de la jerárquica
- ▶ Organizado en anillos concéntricos o ring
 - cada uno tiene una apertura
 - puerta o trampa (trap)
 - por donde pueden entrar las llamadas de las capas inferiores.
 - las zonas más internas del SO o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas
 - Las capas más internas serán, por tanto, más privilegiadas que las externas

Estructura en anillos



Máquina virtual

- ▶ Permite crear sobre una máquina varias máquinas virtuales
- ▶ Presenta una interfaz a cada proceso
 - mostrando una máquina que parece idéntica a la máquina real subyacente.
- ▶ En toda computadora se pueden definir dos máquinas abstractas
 - **M. desnuda**
 - Definida por el hardware
 - Operaciones en lenguaje máquina
 - **M. extendida**
 - se constituye a partir de la máquina desnuda
 - Definida por el hardware y el núcleo

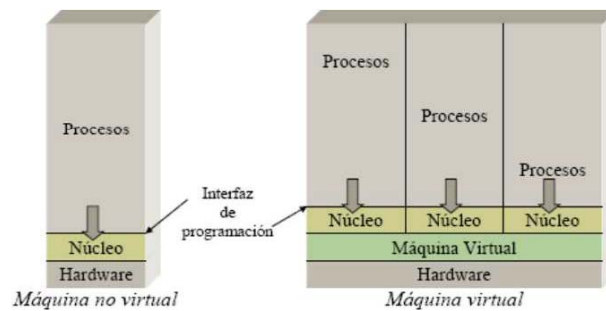


Máquina virtual

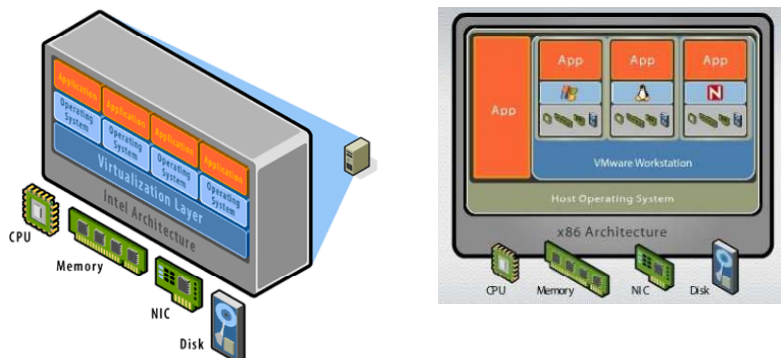
- ▶ El núcleo de estos SSOO se denomina **monitor virtual**
 - lleva a cabo la multiprogramación entre las MV
 - El SO crea la ilusión de múltiples procesos, cada uno de ellos ejecutando su propio procesador con su propia memoria virtual
 - presentando a los niveles superiores tantas MMVV como se soliciten

Máquina virtual

- ▶ Máquina virtual
 - Replica de la máquina real
 - no m. extendida
 - Cada máquina un SSOO distinto → m. extendida al usuario



VMWare vSphere vs VMWare Player o similar

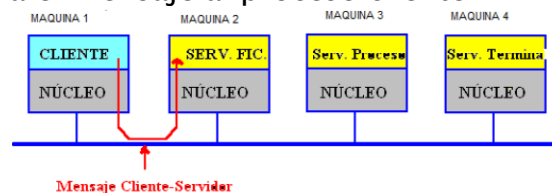


Estructura Cliente-Servidor

- ▶ El tipo más reciente de SO
- ▶ Puede ser ejecutado en la mayoría de las computadoras
 - grandes o pequeñas.
- ▶ De propósito general
 - toda clase de aplicaciones
- ▶ Altamente modular
 - Módulos sin acceso al hardware
- ▶ El núcleo tiene como misión establecer la comunicación entre los clientes y los servidores.
- ▶ Los procesos pueden ser tanto servidores como clientes.
 - Un SO puede ser servidor y cliente a la vez
- ▶ Estos servidores deben tener mecanismos de seguridad y protección
 - son filtrados por el núcleo que controla el hardware.

Estructura Cliente-Servidor

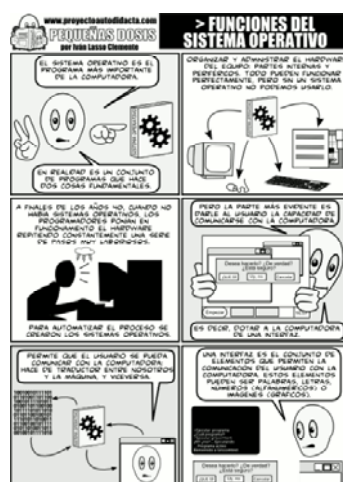
- ▶ Técnica *message passing*
 - El proceso cliente solicita al núcleo un servicio mediante un mensaje
 - El núcleo recibe el mensaje, envía el mensaje al servidor
 - El proceso servidor ejecuta la función solicitada
 - Devuelve un mensaje al núcleo con el resultado
 - El núcleo reenvía el mensaje al proceso cliente



Arquitectura de un SO

- ▶ Un SO
 - es un programa o conjunto de programas
 - que actúa como intermediario entre el usuario y el hardware del ordenador
 - gestionando los recursos del sistema y optimizando su uso.
- ▶ El SO es en sí mismo un programa, pero un programa muy especial y quizá el más complejo e importante.
- ▶ El SO presenta al usuario la máquina de una forma más fácil de manejar y programar que el hardware que está por debajo
 - un usuario normal
 - → abre los ficheros que grabó en un disco
 - → no se preocupa
 - por la disposición de los bits en el medio físico
 - los tiempos de espera del motor del disco
 - la posición de un cabezal el acceso de otros usuarios, etc.

Arquitectura de un SO



Evolución de los SSOO

- ▶ Década de 1940
- ▶ Sin SO
 - Interactúan directamente con el HW
- ▶ Ordenadores de gran tamaño (ENIAC 1946 - 180m²)
- ▶ Programas en código máquina, directamente con el HW
- ▶ Comunicación con la máquina:
 - Panel de programación (displays y switches)
 - Dispositivos de entrada: consola con interruptores manuales
 - Dispositivos de salida: Bombillas de luz

Evolución de los SSOO

- ▶ ENIAC



Evolución de los SSOO

- ▶ Década de 1950
- ▶ Se diseñaron nuevos SSOO
 - hacer más fluida la transición entre trabajos
 - se perdía demasiado tiempo entre la finalización de un trabajo y el comienzo de otro
- ▶ Cuando el trabajo estaba en ejecución
 - tenía control total sobre la máquina
- ▶ Al finalizar el trabajo
 - el control era devuelto al SO
 - mostraba los resultados
 - comenzaba el trabajo siguiente.

Evolución de los SSOO

- ▶ Aparecen nuevos dispositivos de entrada y salida:
 - Tarjetas perforadas e Impresoras
- ▶ Aparecen los **cargadores**, los primeros lenguajes y librerías comunes
- ▶ Comunicación con la máquina:
 - Dispositivos de entrada: lector de tarjetas con el programa
 - Dispositivos de salida: impresora
 - Dispositivo de gestión: consola
- ▶ Los programas eran cargados manualmente en la memoria por el operario (tarjetas)
- ▶ La activación de los programas y recogida de datos se realizaba directamente desde la memoria del ordenador mediante una consola



Procesamiento en Serie

- ▶ Los usuarios acceden en serie a la máquina
 - Uno detrás de otro
- ▶ Ventajas:
 - Interactivo (los usuarios obtienen respuesta inmediata)
- ▶ Desventajas:
 - Sistema monousuario/monopuesto
 - Los usuarios acceden en serie a la máquina. Solamente un usuario operando la máquina en cada momento.
- ▶ Máquina cara y permanece bastante tiempo ociosa
 - debido a que las personas son lentas.
- ▶ Programación & depuración tediosas
 - Cada programa debe incluir código para operar periféricos: propenso a errores.

Procesamiento en Serie

- ▶ Problemas principales
 - Planificación:
 - Formularios de reserva EN PAPEL (p.e: múltiplos de 30 min.)
 - Desperdicio del tiempo del computador
 - Tiempo de preparación:
 - trabajo → compilador → programa objeto → montaje → carga → ejecución
 - Cada paso podía implicar montar y desmontar cintas y/o tarjetas
- ▶ Gran pérdida de tiempo en la preparación
- ▶ Si se produce un error el usuario debía comenzar el proceso

Horario	Nombre trabajo	Usuario
10:00-10:30	Trabajo i	Usuario i
10:30-11:45	-----	-----
11:45-12:30	Trabajo j	Usuario j
12:45-13:30	Trabajo k	Usuario k

Sistema por Lotes

► Problema:

- Máquina antiguas muy caras y permanecían bastante tiempo ociosa, debido a que las personas son lentas.
- Necesidades de maximizar utilización (evitar perdidas en planificación y preparación)

► Solución:

- Nuevo concepto software: **monitor**
- El usuario no accede directamente a la máquina
- Un **trabajo** es un lote de tarjetas perforadas por el programador, posteriormente se utilizaban cintas magnéticas.

Sistema por Lotes

► Los primeros sistemas por lotes se desarrollaron:

- 1940–1950, General Motors, en un IBM701
- 1960, IBM, IBSYS en un 7090/7094

► Operativa:

- El usuario entrega los trabajos al operador
- El operador agrupaba trabajos en un lote y **el monitor cargaba trabajos** y los ejecutaba continuamente
- Los trabajos tenían unas rutinas finales que devolvían el control al monitor cuando terminaban

Sistema por Lotes

- ▶ El monitor debe estar siempre en memoria:
 - **Monitor Residente**
- ▶ Ofrece también un conjunto de funciones comunes como subrutinas para los programas de usuario
 - El **monitor** lee trabajos uno a uno
 - El control pasa al trabajo
 - Al terminar, el trabajo devuelve el control al monitor
 - El resultado de cada trabajo es impreso
 - El **monitor** lee un nuevo trabajo
 - Si ocurre alguna condición de error, el monitor también retoma el control

Sistema por Lotes

- ▶ **Ventajas:**
 - Computador se mantiene la mayor parte del tiempo ocupado
- ▶ **Desventajas:**
 - No interactivo, prolongados tiempos de despacho
 - Procesador costoso y aún permanece ocioso debido a trabajos limitados por E/S.
 - Cantidad de memoria ocupada por el monitor
 - Las desventajas son mejoras deseadas, aun con esto la mejora respecto a los sistemas anteriores son evidentes.

Sistema por Lotes. Características

- ▶ **Protección de memoria:**
 - mientras el programa del usuario este ejecutándose, no debe modificarse la zona de memoria en la que esta el monitor.
 - Error → Control al monitor → siguiente trabajo
- ▶ **Uso de temporizador:**
 - impide que un solo trabajo monopolice el sistema.
 - El temporizador se carga al comenzar cada trabajo y, si expira el tiempo, se producirá una interrupción y el control volverá al monitor.
- ▶ **Instrucciones privilegiadas:**
 - ciertas instrucciones son designadas como privilegiadas y pueden ser ejecutadas solo por el monitor.
- ▶ **El tiempo de maquina:**
 - se reparte entre la ejecución de programas de usuario y la ejecución del monitor.

Sistema por lotes con multiprogramación

- ▶ Hasta la mitad de la década de los 60
- ▶ Máquinas muy caras y la ocupación de la CPU es pequeña por las esperas de E/S
- ▶ **Objetivo:**
 - Disminuir el tiempo de espera de la CPU ejecutando simultáneamente varias tareas
- ▶ **Solución:**
 - Mientras una tarea espera E/S otra tarea puede ejecutarse en el procesador
- ▶ Si hay espacio en memoria para: Monitor + 2 o más prog → Programa 1 (ops. E/S) → Programa 2 (paso a ejecución)

Multiprogramación

- ▶ Es el punto central de los Sistemas Operativos Modernos
- ▶ El **monitor residente** comienza a llamarse **Sistema Operativo**
- ▶ **Características del S.O. necesarias:**
 - Gestión de Memoria
 - Planificación de procesos

Sistemas de tiempo compartido

- ▶ Además de la multiprogramación los sistemas por lotes necesitan interactuar con el usuario.
- ▶ Varias tareas interactivas ejecutándose a la vez
 - Cada usuario accede al sistema mediante terminales.
 - Dedicar a la tarea de cada usuario un **quantum** de tiempo, alternando los programas de estos.
- ▶ Se conoce como **Tiempo compartido**

Sistemas distribuidos

► Procesamiento centralizado

- Un sistema de procesamiento de datos
- en que todas las funciones estan centralizadas en una CPU
- y en un SO

► Procesamiento distribuido

- se ejecutan los datos en distintos nodos, dispersos geograficamente, interconectados mediante una red.

Sistemas distribuidos

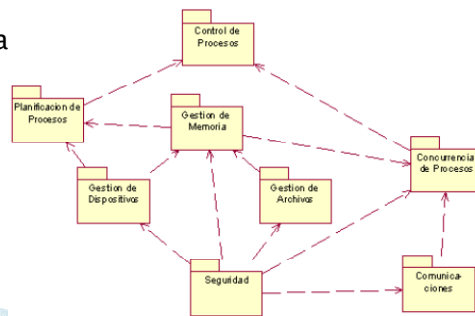
► Características:

- Fragmentación de los elementos que componen una aplicación
 - en dos o más sistemas interconectados, de igual o diferente arquitectura operativa.
- Los recursos de los sistemas se controlan y administran en forma independiente
- La relación entre ambos sistemas puede tener diferentes formas:
 - arquitectura cliente/servidor
 - punto a punto (ambos nodos ofrecen los mismos servicios).

Funciones del SO

► Funciones principales que realiza todo sistema operativo:

- Control de la ejecución de programas
- Administración de periféricos
- Gestión de permisos y de usuarios
- Control de concurrencia
- Control de errores
- Administración de memoria
- Control de seguridad



Clasificaciones de los SSOO

► Por los servicios ofrecidos

- Por el número de usuarios
 - Monousuario.
 - Multiusuario.
- Por el número de tareas
 - Monotarea.
 - Multitarea.
- Por el número de procesadores
 - Monoproceso
 - Multiproceso
 - Simétricos
 - Asimétricos

► Por la forma de ofrecer los servicios

- Sistemas centralizados
- Sistemas de Red
- Sistemas distribuidos
- Sistemas operativos de escritorio

► Por su disponibilidad

- SO propietarios
- SO libres

Por los servicios ofrecidos

- ▶ Por el número de usuarios
 - Monousuario.
 - Multiusuario.
- ▶ Por el número de tareas
 - Monotarea.
 - Multitarea.
- ▶ Por el número de procesadores
 - Monoproceso
 - Multiproceso
 - Simétricos
 - Asimétricos

Por el número de usuarios

- ▶ Monousuario.
 - Únicamente soportan un usuario a la vez
 - sin importar las características de la máquina sobre la que es montado el sistema.
 - MS-DOS, Windows cliente o home
- ▶ Multiusuario.
 - Son capaces de dar servicio a más de un usuario a la vez
 - también independientemente de la plataforma *hardware*
 - A través de terminales
 - UNIX
 - Cliente/ Servidor
 - Windows

Por el número de tareas

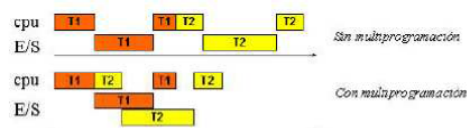
► Monotarea.

- Son sistemas antiguos (asociados a SO monolíticos)
- solo permiten que un programa acapare al procesador o la memoria.
- Sólo permiten una tarea a la vez por usuario
- Puede darse el caso de un sistema multiusuario y monotarea
 - se admiten varios usuarios al mismo tiempo
 - pero de todos puede estar sólo una tarea al mismo tiempo

Por el número de tareas

► Multitarea.

- Este tipo de SO permiten la ejecución de varios programas a la vez.
- Le permite al usuario estar realizando varios trabajos al mismo tiempo.
- Es común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón
 - permite un rápido intercambio de tareas para el usuario, mejorando su productividad.
- La multitarea es relativamente falsa. Multitarea Virtual
 - Un micro solo puede ejecutar una instrucción cada vez



Por el número de procesadores

► Monoproceso

- Permiten utilizar un único procesador
- Permiten simular la multitarea
 - El sistema realiza una tarea rotatoria con intercambio muy rápido.

Por el número de procesadores

► Multiproceso

- permiten utilizar varios procesadores simultáneamente
 - son capaces de ejecutar varias tareas al tiempo.
- Ventajas:
 - Pueden ejecutar varias instrucciones simultáneamente (en paralelo).
 - Aumento del rendimiento (más trabajos en menos tiempo).
 - Compartición de periféricos y fuentes de potencia.
 - Tolerancia a fallos (degradación gradual).
- Desventaja:
 - Sincronización entre procesos.
- Simétricos (SMP, Symetrical Multiprocessing)
 - distribuyen la carga de procesamiento por igual entre todos los procesadores existentes.
- Asimétricos (AMP, Asymetrical Multiprocessing)
 - Determinados procesos los ejecutará siempre un procesador
 - Otro procesador sólo se utilizará para realizar procesos o programas de usuario
 - Es posible que un procesador esté siempre trabajando y el otro sin actividad

Por la forma de ofrecer los servicios

- ▶ Sistemas centralizados
- ▶ Sistemas distribuidos
- ▶ Sistemas de Red
- ▶ Sistemas escritorio

Sistemas Centralizados

- ▶ Los ordenadores personales no tenían
 - un precio asequible y suficiente potencia
 - Sistemas (UNIX)
 - modelo de proceso centralizado.
- ▶ *Mainframe* se encarga de todo el procesamiento
- ▶ Los usuarios manejaban únicamente terminales "tontos"
 - no disponían de memoria, ni de procesador

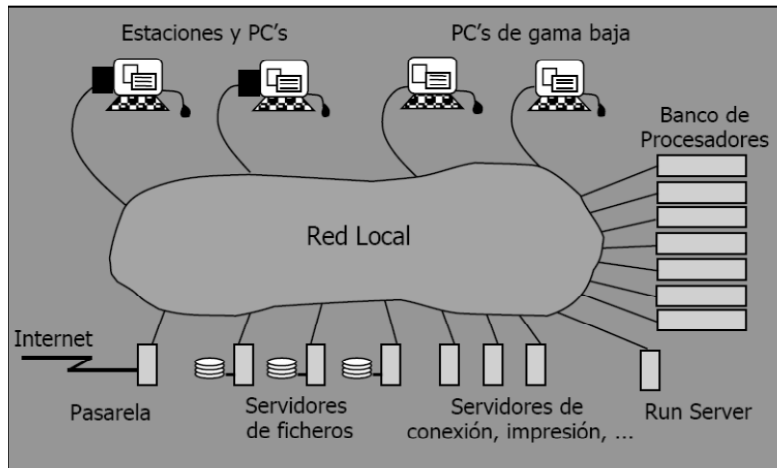
Sistemas Centralizados

- ▶ Actualmente se siguen utilizando los sistemas centralizados
 - *Terminal Services* de Microsoft
 - los terminales dejan de ser tontos
 - pueden realizar otras muchas tareas por sí mismos.

Sistemas distribuidos

- ▶ sistemas independientes
 - permiten distribuir los trabajos, tareas o procesos
 - entre un conjunto de procesadores.
 - en el mismo equipo
 - en equipos distintos
 - están conectados a través de una red de comunicaciones
 - transparente para el usuario
- ▶ Los usuarios desconocen que se trata de un sistema centralizado

Sistemas distribuidos



Sistemas de Red

- ▶ Tienen a dos o más computadoras unidas a través de algún medio de comunicación (físico o no)
- ▶ Comparten los diferentes recursos y la información del sistema.
- ▶ Cada ordenador mantiene
 - su propio sistema operativo
 - su propio sistema de archivos local.
- ▶ Los SSOO de red usados más ampliamente son:
 - Windows Server, Linux Server, etc.

Sistemas de Red vs Distribuidos

- ▶ SO de Red
 - Los usuarios saben de la existencia de varias computadoras pueden
 - conectarse con máquinas remotas
 - copiar archivos de una máquina a otra.
 - Cada máquina
 - ejecuta su propio SO local
 - Se ejecuta de manera independiente de otras máquinas en la red.
 - tiene su propio usuario o grupo de usuarios
 - Esquema de comunicaciones.
- ▶ SO Distribuido
 - Aparece ante sus usuarios como un sistema tradicional de un solo procesador
 - Parece un solo SO que controla la red.
 - Los usuarios no deben saber
 - el lugar donde su programa se ejecuta
 - el lugar donde se encuentran sus archivos
 - Es indispensable el uso de redes para intercambiar datos.

SSOO de escritorio

- ▶ Se utilizan en los equipos de sobremesa, estaciones de trabajo o portátiles.
- ▶ También se les puede denominar como **sistemas operativos cliente**.
- ▶ Entre ellos se encuentran: Windows Vista, Windows 7, Windows 8, Windows 10 y Linux.

Por su disponibilidad

- ▶ SSOO propietarios
- ▶ SSOO libres

SSOO propietarios

- ▶ Son propiedad intelectual de alguna empresa.
 - Se necesitan licencias de uso para que el usuario ejecute el software
 - No tiene acceso a su código fuente
 - Si se puede acceder a el
 - no se puede modificarlo ni distribuido.
 - En este grupo se encuentra Windows.

SSOO Libres

- ▶ Son aquellos que garantizan las cuatro libertades del software libre (según Richard M. Stallman)

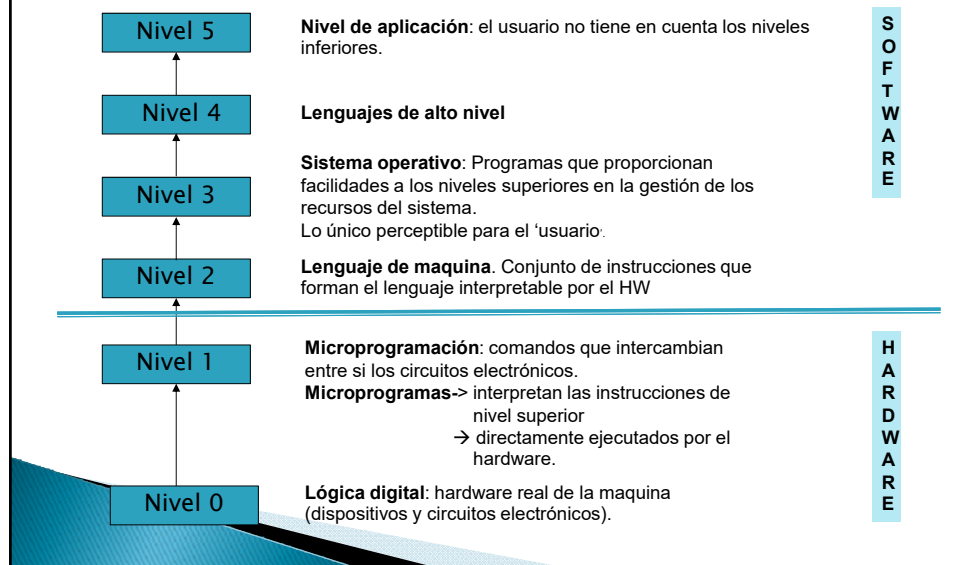


- ▶ Las cuatro libertades de los usuarios del software:
 - La libertad de **usar el programa**, con cualquier propósito
 - libertad 0
 - La libertad de **estudiar como funciona** el programa, y **adaptarlo** a tus necesidades
 - libertad 1
 - El acceso al código fuente es una condición previa para esto.
 - La **libertad de distribuir** copias, con lo que puedes ayudar a tu vecino
 - libertad 2
 - La **libertad de mejorar el programa** y **hacer publicas las mejoras** a los demás, de modo que toda la comunidad se beneficie.
 - libertad 3
 - El acceso al código fuente es un requisito previo para esto.

Tipos de software

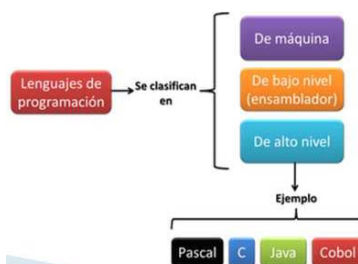
- ▶ Software:
 - como el conjunto de instrucciones o programas usados por una computadora para hacer determinada tarea.
 - Virtual, intangible y están almacenados en diferentes sistemas de almacenamiento.
 - Relación hardware \Leftrightarrow software
 - visión global de la estructura de un ordenador
 - sistema completo esta formado por
 - subsistemas relacionados entre si de forma escalonada.
 - Niveles inferiores: la frontera entre el hardware y el software es bastante difusa.

Subsistemas del ordenador



Lenguajes de programación

- ▶ **Instrucciones:**
 - distintas órdenes para operar sobre los datos
- ▶ **Programa:**
 - conjunto ordenado de **instrucciones**
 - ejecutadas en secuencia
 - con eventuales cambios de flujo causados por el propio programa o eventos externos.
- ▶ **Lenguaje de programación:**
 - conjunto de símbolos y reglas para codificar las **instrucciones**



Lenguajes de programación

► Lenguaje maquina

El **lenguaje máquina** son las instrucciones nativas de un **procesador** en particular

Los lenguajes de máquina son dependientes de la máquina

```
0010 0001 1010
1010 1011 1000
0110 1101 0001
```

Computadora

Lenguaje de máquina

Hardware

Por lo general los lenguajes máquina consisten en cadenas de números (0s y 1s) que instruyen a las computadoras para realizar sus operaciones más relevantes



```
010100100011101011001110
110101001110101010001010
011001110101110101110110
```



Lenguajes de programación

► Lenguajes de bajo nivel

- son mas fáciles de utilizar que los lenguajes máquina
- dependen de cada microprocesador
- El lenguaje de bajo nivel por excelencia es el **ensamblador**
- Las instrucciones en lenguaje ensamblador son instrucciones conocidas como nemotécnicos
- Nemotécnicos típicos
 - de operaciones aritméticas son: en ingles, ADD, SUB, DIV, etc.;
 - de salto: jmp
 - de asignación MOV
- Las instrucciones están formadas por dos campos:
 - **Código de operación:**
 - indica la operación a realizar
 - **Operando:**
 - indica la direccion en memoria donde se encuentran los operandos o el valor de dichos operandos
- JMP 25Hx
- MOV DX, DAT01
 - Dx ← 25Hx

Lenguaje ensamblador

- ▶ Los programas **traductores** (**ensambladores**) convierten el **código fuente**...



- ▶ Escritos en **lenguaje ensamblador a lenguaje máquina**, traduciendo las **instrucciones mnemónicas** a su equivalente en lenguaje máquina

Lenguajes de alto nivel

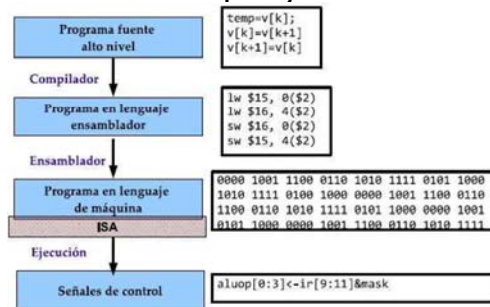
- ▶ Son los mas utilizados por los programadores.
- ▶ Están diseñados para que las personas escriban y **entiendan** los programas de un modo mucho mas fácil que los lenguajes maquina y ensambladores.
- ▶ Un programa escrito en lenguaje de alto nivel es **independiente de la maquina**
 - Las instrucciones del programa no dependen del diseño del hardware o de un ordenador en particular.
 - Los programas escritos en lenguaje de alto nivel son portables



Lenguajes de programación

► Compilador

- traductor de un **programa fuente** que se encuentra en un **lenguaje de alto nivel**, para producir un **programa objeto** en un lenguaje de bajo nivel (**ensamblador o código máquina**).
- Tiene como objetivo obtener un **programa ejecutable**.



Software libre

► El software libre

- Las **cuatro libertades de Stallman**
- Suele estar disponible gratuitamente
 - o al precio de coste de la distribución a través de otros medios
 - no es obligatorio que sea así
 - No hay que asociar software libre a **software gratuito**, ya que, conservando su carácter de libre, podrá ser distribuido comercialmente (**software comercial**).
- El **software gratuito (freeware)**
 - puede incluir el código fuente
 - eso no quiere decir que se pueda considerar como "libre" a no ser que se garanticen los derechos de modificación y redistribución de las versiones modificadas del programa.

Software de dominio público

- ▶ Tampoco debe confundirse software libre con **software de dominio publico**.
 - tiene como particularidad la **ausencia de Copyright**
 - es software libre sin derechos de autor.
 - En este caso los autores renuncian a todos los derechos que les puedan corresponder.

Tipos de aplicaciones

- ▶ Gratuitas o Comerciales (de pago)
 - **Shareware**
 - evaluar de forma gratuita el producto, pero con limitaciones de tiempo o de capacidades

Tipos de aplicaciones

▶ Libres o propietarias

- Libre
 - Distribución del código fuente junto con el programa
 - cuatro premisas Stallman
 - Gratuito o no
- Propietario
 - Usuarios tienen limitadas las posibilidades:
 - licencia
 - de usarlo
 - modificarlo
 - redistribuirlo (con o sin modificaciones).

Tipos de aplicaciones

▶ Opensource

- código abierto al usuario

▶ Privativas

- código fuente no está disponible o el acceso a él se encuentra restringido

Actividad 2.1

- » Cual es el tipo de aplicación de Adobe Read

Actividad 2.2

- » Busca aplicaciones de los distintos tipos

Tipos de licencia de distribución

- ▶ OEM
- ▶ Retail
- ▶ Licencias por volumen

Licencia OEM

- ▶ Licencia que supeditada a la compra de un equipo nuevo.
- ▶ Prohibida la venta por separado
- ▶ Sobre todo en SSOO
- ▶ Aunque sea propiedad del comprador, puede haber limitaciones de uso
 - número máximo de veces que se puede reinstalar.
- ▶ NO se pueden vender, ni ceder a terceros
 - salvo en las mismas condiciones en las que se compraron
 - como parte de un equipo

Licencia Retail

- ▶ Retail (venta)
 - Venta de software
 - Propiedad del usuario
 - Pudiendo cederlo libremente a terceros o venderlo.

Licencias por volumen

- ▶ Destinado a grandes empresas
 - Condiciones similares a las de las licencias OEM
 - Sin equipos nuevos.
 - Número de equipos
 - mismo código de licencia
- ▶ Fabricante esta autorizado para hacer las comprobaciones de numero
- ▶ NO se puede ceder a terceros ni total ni parcialmente

Gestores de arranque

- ▶ Arranque de un Sistema
 - POST
 - Se ejecuta el MBR
 - MBR busca la partición activa
 - Ejecuta su PBR (Partition Boot Record)
 - Llama al gestor de arranque del SO
- ▶ Gestor de arranque
 - pequeño programa
 - permite seleccionar el SO en caso de disponer de arranque múltiple.
- ▶ Gestores
 - NTLDR
 - Bootmgr
 - Lilo
 - Grub

Gestores de arranque

- ▶ **Lilo** (Linux Loader)
 - Gestor de arranque de Linux
 - Permite iniciar Windows
 - Múltiples sistemas de archivos
 - Arrancar un SO desde el disco duro o desde un disco externo
- ▶ **Grub**
 - Mas moderno y flexible que Lilo
 - Permite que el administrador ejecute cualquier comando desde la línea de comando de Grub.
 - Características
 - Posibilidad de incluir múltiples formatos de ejecutables
 - Puede arrancar SO no-multiarranque
 - Agradable interfaz de usuario
 - Interfaz de línea de comando muy flexible.

Actividad 2.4

- » Averigua qué gestor de arranque utiliza el sistema operativo de su equipo.

Gestores de arranque vs Máquinas virtuales

- ▶ Gestor de arranque
 - ahorro de hardware
 - No se puede disponer de ambos los SSOO simultáneamente.
 - Solución Máquinas Virtuales
- ▶ Máquinas Virtuales
 - Varios SSOO a la vez
 - Pueden interactuar