Unidad 1: Conceptos generales de Sistemas Operativos.

Tema 3: Estructura del sistema operativo.

3.1 Componentes del sistema.

3.2 Servicios del sistema operativo.

3.3 Llamadas al sistema.

3.4 Programas del sistema.

3.5 Estructura del sistema.

3.6 Máquinas virtuales.

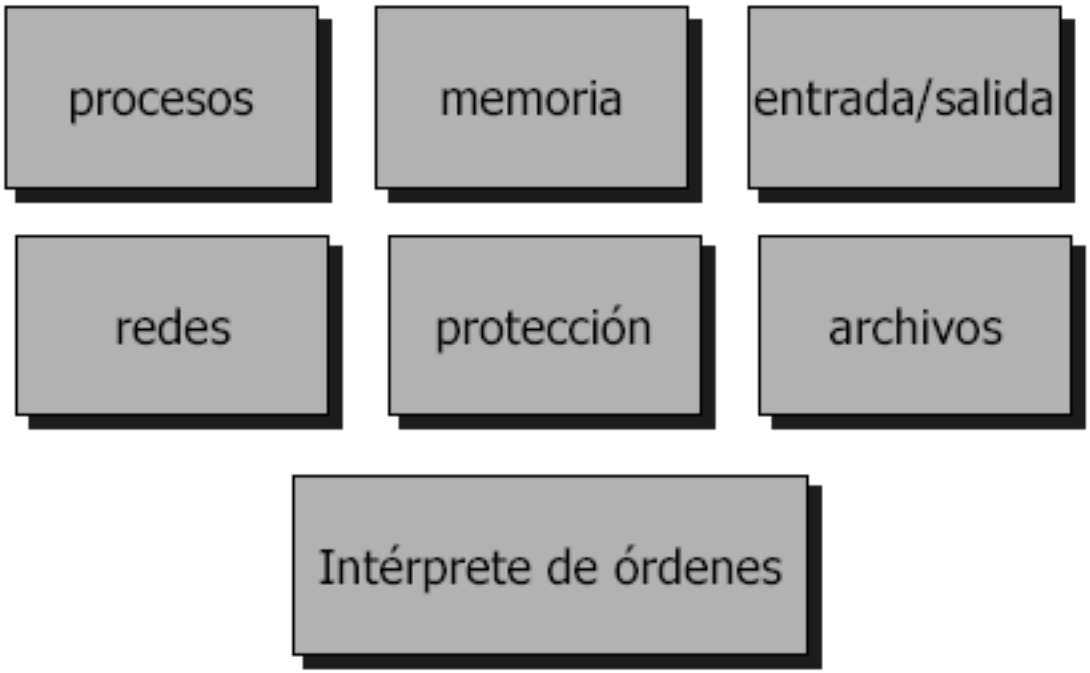
3.7 Modelo cliente – servidor.

3.8 Diseño e implementación de sistemas.

3.9 Generación de sistemas.

3.1 Componentes del sistema:

- **Sistema grande y complejo:**
  - División en componentes, con sus entradas, salidas y funciones cuidadosamente definidas:



3.1 Componentes del sistema:

- **3.1.1 Gestión de procesos:**
  - **Proceso:** programa en ejecución.
  - **Necesita** de ciertos **recursos**:
    - Tiempo de CPU.
    - Memoria.
    - Archivos.
    - Dispositivos de E/S.
    - Puede necesitar también datos de inicialización.
  - A la **finalización de un proceso** el SO **recupera los recursos** que le había asignado.
  - Es la unidad de trabajo de un sistema. El sistema consiste sólo en una colección de procesos.

3.1 Componentes del sistema:

- **3.1.1 Gestión de procesos (2):**
  - El SO se encarga de las siguientes actividades relacionadas con la gestión de procesos:
    - Crear y eliminar procesos.
    - Suspender y reanudar procesos.
    - Proveer mecanismos para la sincronización de procesos.
    - Proveer mecanismos para la comunicación de procesos.
    - Proveer mecanismos para manejar bloqueos mútuos.

3.1 Componentes del sistema:

- **3.1.2 Gestión de la memoria principal:**
  - **Memoria principal:** almacén de datos de **acceso rápido**, que son **compartidos** por la CPU y los dispositivos de E/S.
  - Es el único dispositivo de almacenamiento grande que **la CPU puede direccionar y acceder directamente.**
  - Las **instrucciones** deben estar **en la MP** para que la CPU pueda ejecutarlas (es preciso cargar los programas en MP).
  - El SO se encarga de las siguientes actividades relacionadas con la gestión de memoria:
    - Saber qué partes de la memoria se están usando, cuáles están libres y quién las está usando.
    - Decidir **qué procesos cargar en la memoria**.
    - **Asignar y liberar espacio** de memoria.

3.1 Componentes del sistema:

- **3.1.3 Gestión de archivos:**
  - **Archivo:** conjunto de **información relacionada**, generalmente programas y datos. Se organizan en directorios para hacer su uso más sencillo. Cuando varios usuarios tienen acceso a los archivos, se debe controlar quién y de qué modo accede a ellos.
  - El SO se encarga de las siguientes actividades relacionadas con la gestión de archivos:
    - Crear y eliminar archivos.
    - Crear y eliminar directorios.
    - Proveer las primitivas para manejo de archivos y directorios.
    - Establecer la correspondencia archivo-almacenamiento secundario.
    - Guardar los archivos en almacenamientos no volátiles.

3.1 Componentes del sistema:

- **3.1.4 Gestión del sistema E/S:**
  - Se trata de un conjunto de dispositivos muy variados y complejos de programar.
  - El SO se encarga de las siguientes actividades relacionadas con la gestión del sistema E/S:
    - Proporcionar una interfaz uniforme para el acceso a los dispositivos.
    - Proporcionar manejadores para los dispositivos concretos.
    - Tratar automáticamente los errores más típicos.
    - Para los dispositivos de almacenamiento, usar cachés.
    - Para los discos, planificar de forma óptima las peticiones.

3.1 Componentes del sistema:

- **3.1.5 Gestión de almacenamiento secundario:**
  - **Almacenamiento no volátil**, casi todos los programas (compiladores, ensambladores, rutinas de ordenación, editores y formateadores) se guardan en disco hasta que se cargan en memoria.
  - El SO se encarga de las siguientes actividades relacionadas con la gestión de discos:
    - Administración del espacio libre.
    - Asignación del almacenamiento.
    - Planificación del disco.

3.1 Componentes del sistema:

- **3.1.6 Trabajo con redes:**
  - **Sistema distribuido:** colección de procesadores con sus propios recursos locales (memoria local, reloj) y que se comunica con otros procesadores conectados mediante una red.
  - Objetivos del SO:
    - Proporcionar primitivas (de comunicación) para **conectarse con equipos remotos** y acceder de forma controlada a sus recursos.

3.1 Componentes del sistema:

- **3.1.7 Sistema de protección:**
  - Es preciso proteger a un proceso de los demás, ya que el sistema de computación admite **múltiples usuarios** y la **ejecución concurrente** de procesos.
  - **Protección:** Mecanismo para **controlar el acceso** de programas, procesos o usuarios a los recursos definidos por un sistema de computador.
  - La protección puede **mejorar la confiabilidad** mediante la detección de errores.

3.1 Componentes del sistema:

- **3.1.8 Sistema de interpretación de órdenes:**
  - **Interfaz entre usuario y sistema operativo**. Para que un usuario pueda **dialogar** directamente con el SO, se proporciona una interfaz de usuario básica para:
    - Cargar programas.
    - Abortar programas.
    - Introducir datos a los programas.
    - Trabajar con archivos.
    - Trabajar con redes.
  - Ejemplos: JCL en sistemas por lotes, COMMAND.COM en MS-DOS, shell de UNIX.

3.2 Servicios del sistema operativo:

- **Servicios a los programas y a sus usuarios:**
  - Ejecución de programas.
  - Operaciones de E/S.
  - Manipulación del sistema de archivos.
  - Comunicaciones: entre procesos y de red.
  - Detección de errores.

- **Asegura el funcionamiento eficiente del sistema:**
  - Asignación de recursos: varios usuarios – varios trabajos.
  - Contabilización: qué usuarios usan qué recursos.
  - Protección: controlar accesos a los recursos.
  - Seguridad: cada usuario debe identificarse.

3.3 Llamadas al sistema:

- Interfaces con los servicios del sistema operativo:
  - Para el programador: llamadas al sistema en lenguaje máquina o en alto nivel.
  - Para el usuario:
    - Intérprete de órdenes.
    - Programas del sistema.
- El SO ofrece una gama de servicios a los programas, que acceden a ellos mediante llamadas al sistema.
- Son la interfaz entre el programa en ejecución y el SO.
- Única forma en la que un programa puede solicitar operaciones al SO.
- Ejemplo de llamadas al sistema:
  - UNIX: fd = open (“mifichero”, O_RDONLY);

3.3 Llamadas al sistema:

- Implementación de las llamadas al sistema:
  - ¿Cómo se implementa la llamada?
    - Habitualmente, mediante una instrucción especial de la máquina (syscall, int, trap, ...).
    - La instrucción cambia automáticamente a modo privilegiado.
    - Si programamos en un lenguaje de alto nivel escribimos la llamada al sistema como una subrutina, y el compilador la sustituye por la instrucción de máquina correspondiente.
  - Muchas llamadas necesitan parámetros ¿cómo los pasamos al SO?:
    - Usando registros de la máquina.
    - En una tabla en memoria principal.
    - Poniéndolos en la pila (stack):

3.3 Llamadas al sistema:

- **Tipos de llamadas al sistema:**
  - **Control de procesos:**
    - Fin, abortar, cargar, ejecutar, crear, finalizar, obtener y establecer atributos, espera, asignar y liberar memoria.
  - **Manipulación de archivos:**
    - Crear y eliminar archivo, abrir y cerrar, leer, escribir, reposicionar, obtener y establecer atributos.
  - **Manipulación de dispositivos:**
    - Solicitar y liberar, leer, escribir, reposicionar, obtener y establecer atributos, conectar y desconectar dispositivos.
  - **Mantenimiento de información:**
    - Obtener y establecer hora, fecha, datos del sistema, atributos de un proceso, archivo o dispositivo.
  - **Comunicaciones:**
    - Crear, eliminar conexiones; enviar y recibir mensajes, ...

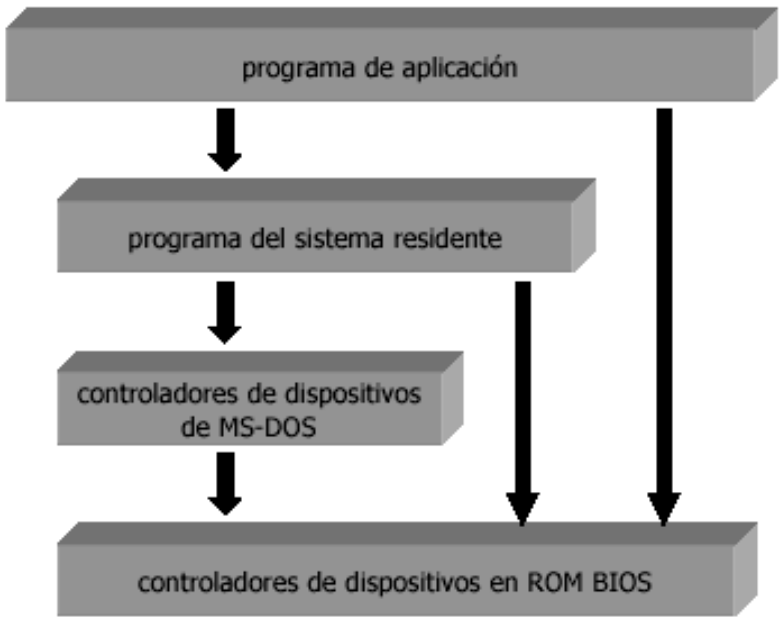
3.4 Programas del sistema:

- El entorno del SO provee de utilidades básicas para:
  - Manipular ficheros.
  - Editar documentos.
  - Proporcionar un entorno de trabajo.
  - Desarrollar programas (compiladores, enlazadores, etc.).
  - Comunicarnos con otros equipos (telnet, ftp, ssh, etc).

- Núcleo (kernel) del SO:
  - Software que **reside permanentemente en memoria** y que atiende las llamadas al sistema y demás eventos básicos.
  - Distinquiremos entre el núcleo y los programas del sistema (que son los que utilizan los servicios del núcleo).

3.5 Estructura del sistema:

- En principio se pensó en una estructura que proporcionara máxima funcionalidad en el mínimo espacio, por lo que no había una buena separación entre interfaces y niveles de funcionalidad.



3.5 Estructura del sistema:

- Estructura del sistema UNIX:
  - 2 partes separables: el núcleo (que se divide en interfaces y controladores de dispositivos) y los programas del sistema.

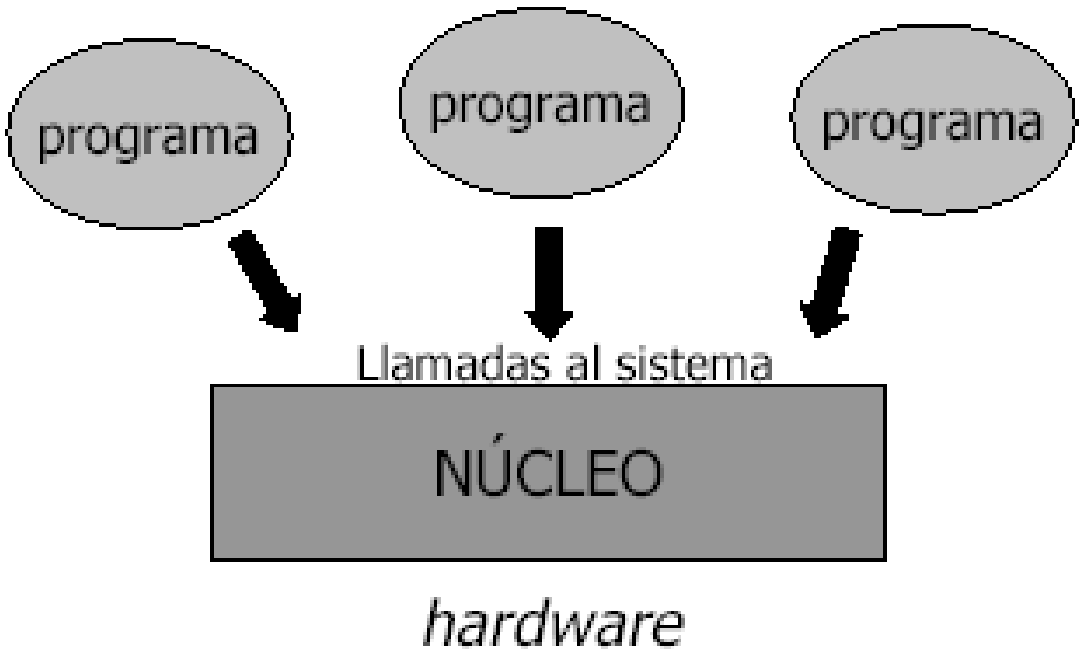
<table>
<tr><td colspan="3">(usuarios)</td></tr>
<tr><td colspan="3">shells y órdenes
compiladores e intérpretes
bibliotecas del sistema</td></tr>
<tr><td colspan="3">Interfaz con el núcleo mediante llamadas al sistema</td></tr>
<tr><td>manejo de terminales por señales
sistema de E/S por caracteres
drivers de terminales</td><td>sistema de archivos
sistema de E/S por intercambio de bloques
drivers de disco y cinta</td><td>planificación de CPU
reemplazo de páginas
paginación por demanda
memoria virtual</td></tr>
<tr><td colspan="3">Interfaz con el núcleo</td></tr>
<tr><td>controladores de terminales
terminales</td><td>controladores de dispositivos
discos y cintas</td><td>controladores de memoria
memoria física</td></tr>
</table>

3.5 Estructura del sistema:

- En su interior, un SO posee una cierta estructura, una organización:
  - Bloque único y sólido de servicios (**sistemas monolíticos**).
  - Serie de capas de software delimitadas y jerarquizadas (**sistemas por capas**).
  - Modelo de **máquinas virtuales**.
  - Modelo **cliente-servidor**.

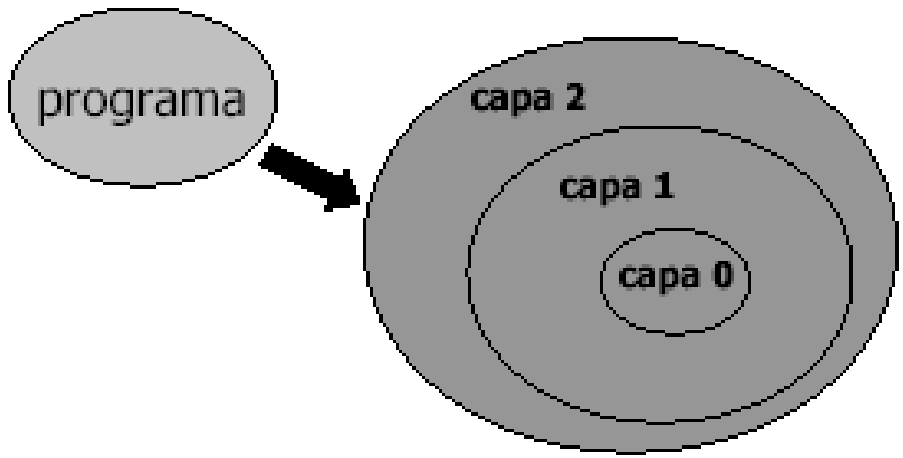
3.5 Estructura del sistema:

- Sistemas monolíticos:
  - La arquitectura más simple para un SO es un **núcleo compacto**, que contiene todas las rutinas del SO.



3.5 Estructura del sistema:

- Sistemas por capas:
  - Sistema **construido según niveles jerárquicos** (capas), aprovechando siempre los servicios de la capa inferior. La capa inferior (capa 0) es el hardware, la capa superior (capa N) es la interfaz con el usuario.
  - Diseño más **modular y escalable** que el monolítico.



3.5 Estructura del sistema:

- Sistemas por capas:
  - Ventajas: **Modularidad.**
    - **Depuración y verificación:** Tras depurar la 1ª capa se supone su funcionamiento correcto mientras se trabaja con la 2ª capa.
    - **Mantenimiento:** Es posible cambiar las rutinas de bajo nivel siempre que la interfaz externa de la rutina no cambie y la rutina realice la misma tarea anunciada.
  - Desventajas: **Definir adecuadamente las distintas capas.**
    - Tienden a ser menos eficientes:
      - Llamadas entre capas => paso de parámetros.
      - Cada capa implica un gasto extra.
      - Tendencia: equilibrio, menos capas con más funcionalidad,
        - Ventajas de la modularidad.
        - Evitan los problemas de definición e interacción entre capas.

3.6 Máquinas virtuales:

- Mediante software, se proporciona a los programas la emulación de un hardware que no existe.
- El software emulador convierte las peticiones hechas a la máquina virtual en operaciones sobre la máquina real.
- Se pueden ejecutar varias máquinas virtuales al mismo tiempo (ej. mediante tiempo compartido).
- Los recursos reales se reparten entre las distintas máquinas virtuales.

3.6 Máquinas virtuales:

- Ejemplos de máquinas virtuales:
  - IBM VM: ofrecía a cada usuario su propia máquina virtual monotarea; las máquinas virtuales se planifican con tiempo compartido.
  - Java: los programas compilados en Java corren sobre una máquina virtual (JVM).
  - VMWare: en un PC, es capaz de ejecutar al mismo tiempo varias sesiones Windows, Linux, OS/2, etc.
  - Nachos: SO que se ejecuta en una máquina virtual MIPS, cuyo emulador corre sobre UNIX.

3.6 Máquinas virtuales:

- Ventajas e inconvenientes de las máquinas virtuales:
  - **Protección:** cada máquina virtual está aislada de las otras y no puede interferir.
  - **Investigación y desarrollo:** se puede desarrollar y ejecutar para un hw que no tenemos.
  - **Independencia del hardware** (ej: Java).
  - **Pervivencia de sistemas antiguos** (ej: emuladores, MS-DOS).
  - La **implementación** de la memoria virtual puede ser **compleja y lenta**.

3.7 Modelo cliente – servidor:

- SO como conjunto de módulos autónomos, cada uno de los cuales tiene a disposición del resto una serie de servicios (competencias).
- Módulo como servidor de determinados servicios que atiende las peticiones de otros módulos y que a su vez puede ser cliente de otros módulos.
- Podemos extender este modelo hasta el infinito si consideramos cada módulo del sistema como un conjunto de módulos con relaciones cliente – servidor.
- El modelo jerárquico sólo es un caso particular del modelo cliente – servidor.
- Indicado para SO.

3.8 Diseño e implementación de sistemas:

- Objetivos del diseño:
  - **Definición de objetivos y especificaciones:**
    - **Selección del hardware.**
    - **Tipo de procesamiento**: por lotes, de tiempo compartido, mono/multiusuario, distribuido, de tiempo real, ...
    - Mayor complicación tiene **especificar los requisitos** a cumplir:
      - Metas del usuario:
        - Cómodo y fácil de usar y de aprender, confiable, seguro, rápido.
      - Metas del sistema (diseñar, crear, mantener y operar):
        - Diseño, implementación y mantenimiento fácel, flexible, confiable, libre de errores, eficiente.
  - Gran **complicación** para definir los **requisitos del SO**, que serán diferentes según cada tipo de sistema.
  - **Ingeniería del software**: especificación y diseño.

3.8 Diseño e implementación de sistemas:

- Mecanismos y políticas:
  - Los **mecanismos** determinar **cómo se hace algo**.
  - Las **políticas** deciden **qué se hace**.
  - Ejemplo: un mecanismo para asegurar la protección de la CPU es la construcción de un temporizador; la decisión de a qué intervalo de tiempo se ajustará el temporizador para un usuario en particular es una decisión de política.
  - Es **conveniente separar los mecanismos de las políticas**. Aunque las políticas cambien, es deseable que los mismos mecanismos pueden seguir siendo útiles.

3.8 Diseño e implementación de sistemas:

- Implementación de sistemas:
  - Tradicionalmente los SO se han escrito en lenguaje ensamblador (por eficiencia).
  - Actualidad: uso de lenguajes de alto nivel; UNIX, OS/2 y Windows NT están escritos principalmente en C.
    - Ventajas:
      - Más legible y fácil de entender y depurar.
      - Más transportable.
    - Desventajas:
      - Menor velocidad.
      - Mayor necesidad de almacenamiento.
  - Nota: la programación de un SO tiene la complicación de las pruebas. Para salvarle se usan emuladores.

3.9 Generación de sistemas:

- Hay que determinar los siguientes tipos de información:
  - ¿Qué CPU se usará?.
  - ¿Qué opciones están instaladas?.
  - ¿Qué memoria se tiene?.
  - ¿Con qué dispositivos se cuenta?.
  - ¿Opciones y parámetros a usar?.
- Tras determinar esta información se incluye en el código fuente y se compila el SO.
- El computador se inicia cargando un núcleo, “arranque del sistema”. El programa de arranque (guardado en ROM) localiza el núcleo, lo carga en memoria e inicia la ejecución.