# TEMA 4

# Introducción a los Sistemas Operativos

Profesora
Alba Alejandre Voces

# Índice


1 Sistema Operativo

2 Evolución histórica

3 Clasificación SO

4 Estructura de un SO

5 Funciones de un Sistema Operativo

# Índice


1 **Sistema Operativo**

2 Evolución histórica

3 Clasificación SO

4 Estructura de un SO

5 Funciones de un Sistema Operativo

# Sistema Operativo

Un Sistema Operativo es el conjunto de programas que **controlan, coordinan y dirigen el uso de los recursos hardware** de un sistema informático por las aplicaciones y los usuarios.

- **Objetivo**
  - Proveer un entorno amigable al usuario
  - Ofrecer una serie de servicios a las aplicaciones de usuario
  - Administrar eficientemente los recursos del sistema informático (espacio en memoria, ciclos de ejecución en la CPU, …)
  - Evolucionar para dar respuesta a los avances tecnológicos que se producen en el hardware

# Índice


1 Sistema Operativo

2 **Evolución histórica**

3 Clasificación SO

4 Estructura de un SO

5 Funciones de un Sistema Operativo


- **Inicio de los ordenadores**
  - No tenían sistema operativo
  - Se diseñaban específicamente para realizar un tipo de tarea muy concreto
  - Estaba encuadrado en proyectos científicos y militares
  - Uso de válvulas de vacío (máquinas muy lentas y elevado consumo de energía)
  - Años más tarde, algunas empresas tecnológicas empezaron a producir ordenadores de **propósito general** para su venta a otras empresas.
  - Es entonces cuanto surge la necesidad de un software básico para su uso, el **sistema operativo**

# Década de los 50

- Aparecen los transistores, que se introducen dentro de la arquitectura de los ordenadores, permitiendo ordenadores más pequeños y potentes
- Monitor residente: software encargado de cargar un programa junto con sus datos (un trabajo) en la memoria del ordenador desde un dispositivo (una cinta magnética o un conjunto de tarjetas perforadas) para luego ejecutarlo
- Inconveniente: al terminar un trabajo tardaba mucho en cargar el siguiente
- Solución: procesamiento por lotes o batch (1955). Se grababan un conjunto de trabajos en una cinta o tarjeta perforada y se cargaban para ejecutarse uno tras otro.
- Para aumentar la productividad, se mejoró el sistema para poder cargar un trabajo o dar salida a los datos mientras se ejecutaba otro. A esto se le conoce como procesamiento fuera de línea (off-line)
- La programación se realizaba en lenguaje ensamblador y en FORTRAN

# Década de los 60

- La aparición de los **circuitos integrados (CI)** supuso una mejora consiguiendo un menor tamaño y relación precio/rendimiento
- **Multiprogramación:** varios programas de usuario se encuentran al mismo tiempo en memoria, y la CPU cambia rápidamente de un trabajo a otro cuando el primero queda bloqueado por una operación de E/S
- **Multiprocesamiento:** se utilizan varias CPUs en un solo ordenador, con la finalidad de incrementar la capacidad de procesamiento de la maquina
- **Tiempo compartido:** tiempo del procesador se comparte entre programas de varios usuarios pudiendo ser programas interactivos
- **Tiempo real:** en que los ordenadores fueron utilizados en el control de procesos industriales.

- **Década de los 70**
  - Introducción de la familia de ordenadores **Sistema/360 de IBM**
  - Ordenadores de **propósito general**, es decir, podían ejecutar diversos tipos de aplicaciones
  - Aparece el sistema operativo **UNIX** en los Laboratorios Bell. Sistema operativo portable, multitarea y multiusuario
  - También aparece un ordenador personal, el **Xerox Alto** que tenía su propio sistema operativo.

# Década de los 80

- Aparición de los **circuitos LSI** (integrados a gran escala)
- Tim Paterson (1979) crea su sistema operativo **86-DOS**, que posteriormente pasó a llamarse QDOS (Quick and Dirty Operative System).
- Poco después Bill Gates compra QDOS y lo rebautiza dos veces: **PC-DOS y MS-DOS.**
- 1983 Apple crea el **Apple Lisa System 1** y un año después el Mac OS
- 1985 1º versión de **Windows**, pero como entorno gráfico bajo MS-DOS
- 1987 Andrew S. Tanenbaum crea **MINIX**, un sistema operativo basado en Unix y escrito en lenguaje C. Gracias a el sistema llevó a Linus Torvalds para la creación del **núcleo Linux**.
- En esta época el principal objetivo de los SO era que fuesen más amigables en lugar de mejorar el rendimiento
- A mediados de década se lleva a cabo el crecimiento de las redes de ordenadores, **con sistemas operativos de red y sistemas operativos distribuidos.**

- **Década de los 90**
  - Aparecen los sistemas operativos con **interfaces gráficas de usuarios (GUI).**
  - 1990: aparece Windows v3 y en 1995 Windows 95 de Microsoft. Ambos siguen siendo entornos gráficos para ser ejecutados bajo el sistema operativo MS-DOS
  - 1998: aparece Windows 98 como sistema operativo con GUI para PC.
  - 1990: Richard Stallman crea el sistema GNU de software libre
  - 1991: Linus Torvalds crearía el núcleo Linux
  - 1992: el sistema GNU y el Núcleo Linux se unen formalmente para crear GNU/Linux

- **Actualidad**
  - Los sistemas operativos evolucionan con nuevas interfaces de usuario, como la pantalla táctil para nuevos dispositivos: PC's, portátiles, tablets, smartphones.
  - Aparecen los sistemas operativos en la nube o sistemas operativos en la web (WebOS). Consisten en un escritorio virtual con aplicaciones integradas que permiten al usuario administrar sus datos sin necesidad de instalar aplicaciones. Todo ello mediante una conexión a Internet.

- Ejemplo: Amazon Web Services. Microsoft Azure, Google App Engine

# Índice


1 Sistema Operativo

2 Evolución histórica

3 **Clasificación SO**

4 Estructura de un SO

5 Funciones de un Sistema Operativo

# Tiempo de respuesta

- **Procesamiento por lotes**: ejecuta un conjunto de trabajos (lote), uno detrás de otro, mostrando el resultado al terminar todos los trabajos. El usuario no interviene.
  - Ejemplo: SCOPE (procesamiento científico pesado), EXEC II (orientado a procesamiento académico)
- **Interactivo:** Existen interacción con el usuario. El SO ejecuta un trabajo y cuando necesita un dato interrumpe su ejecución para solicitarlo al usuario. Se utiliza la técnica tiempo compartido, se divide el tiempo de procesador entre varios trabajos
- **Tiempo real:** el tiempo de respuesta está dentro de un plazo definido de antemano que generalmente suele ser bajo. Este tipo de SO se suele utilizar donde el tiempo de respuesta es crítico.
- Ejemplo: tráfico aéreo, monitorización de pacientes, ..

- **Número de usuarios**

- **Monousuario:** solo un usuario puede utilizar el sistema informático, para el que están disponibles todos los recursos.
    - Ejemplo: MS-DOS, Windows 95,
- **Multiusuario:** varios usuarios **simultáneamente** pueden utilizar el sistema informático, compartiendo los recursos entre ellos. Actualmente este tipo de sistemas se emplean especialmente en redes, pero los primeros sistemas multiusuario fueron sistemas centralizados que se compartían a través del uso de múltiples terminales.
    - Ejemplo: Windows Server 2008

- **Número de procesos**
  - **Monotarea:** solamente ejecuta una tarea a la vez. La CPU ejecuta un programa y hasta que no acaba o ejecuta otro.
    - Ejemplo: MS-DOS
  - **Multitarea:** Se pueden ejecutar varios programas a la vez. Si el sistema informático solamente tiene una CPU, el SO dividirá su tiempo entre todos los procesos intentando maximizar el uso de la CPU. Los procesos en ejecución deberán permanecer en memoria principal mientras se ejecuten y utilizarán algún algoritmo de planificación de procesos
    - Ejemplo: Windows 2000, Windows XP

# • Número de procesadores

- **Monoproceso:** solamente puede trabajar con un único procesador, aunque el sistema informático tenga más de un procesador.

  Ejemplo: MS-DOS, Windows 95, Windows 98

- **Multiproceso:** puede trabajar con varios procesadores con lo que podrá ejecutar varios procesos de manera simultánea. Pueden ser:
  - Simétricos: el SO trabaja indistintamente con cualquier procesador
  - Asimétricos: El SO selecciona un procesador al que le da el papel de primordial o maestro y que seleccionara al resto para distribuir el trabajo

  Ejemplo: Windows 2000, Windows XP

# Clasificación SO

- **Trabajo en red**
  - **Centralizados:** El equipo informático no comparte ningún recurso ni utiliza recursos de ordenadores por la red.
  - **En Red:** permiten compartir recursos y conectar varios equipos entre sí dentro de una red de ordenadores
  - **Distribuidos:** funcionan de manera que para el usuario la existencia de la red pasa desapercibida, facilitando el acceso a los recursos en red del sistema.

# Índice


1 Sistema Operativo

2 Evolución histórica

3 Clasificación SO

4 **Estructura de un SO**

5 Funciones de un Sistema Operativo


# Programas de un SO

- **Núcleo o Kernel:** Se encarga de interactuar con el hardware. Es la única parte del SO que tiene acceso directo al hardware.
- **Llamadas al sistema o primitivas:** Funciones que invocan las aplicaciones de usuario para solicitar algún servicio al SO. Cada SO implementa un conjunto propio de llamadas al sistema.
- **Servidor**: programa que utiliza el kernel para suministrar algún servicio a las aplicaciones de usuario. Los servidores se emplean en estructuras microkernel
- **Interrupciones y excepciones**: El SO ocupa una posición intermedia entre los programas de aplicación y el hardware. El hardware también necesita ejecutar código del SO. Para ello:
  - **Interrupciones:** señal que envía un dispositivo de E/S a la CPU para indicar quela operación de la que se estaba ocupando, ya ha terminado.
  - **Excepción:** una situación de error detectada por la CPU mientras ejecutaba una instrucción, que requiere tratamiento por parte del SO
- **Utilidades:** programas para resolver pequeñas tareas de los usuarios, como el navegador de Internet, el explorador de archivos, …

# Estructura de un SO

- **Ventajas estructuración**
  - Facilite la comprensión
  - incremente la portabilidad y la extensión
  - Favorecer su mantenimiento.

- **Principales estructuras**
  - Sistema  Monolítico
  - Sistema  Microkernel
  - Sistema por capas
  - Sistema por módulos

# Estructura de un SO: Sistema Monolítico

- Es la estructura más simple para un SO, para proporcionar una máxima funcionalidad dentro del menor espacio posible
- No tienen una estructura totalmente clara, es decir, sus rutinas y funcionalidades se encuentran agrupados en un solo programa
- El conjunto de procedimientos o rutinas están entrelazadas de tal forma que cada una tiene la posibilidad de llamar a las otras rutinas cada vez que así lo requiera

## • inconvenientes

- Poca fiabilidad. Se ejecuta todo al mismo nivel por lo que lo hace altamente vulnerable.
- Cuando falla un programa se produce une error en todo el sistema
- Si se modifica el hardware, hace falta recompilar el kernel y esto consume tiempo y recursos
- Se resume en tres grandes inconvenientes: el tamaño del núcleo, la falta de extensibilidad y la dificultad de mantenimiento.
- Ejemplos: Unix, MS-DOS, Mac OS (hasta v8.6), Linux, Windows 95,98, 98SE, Me

# Estructura de un SO: Sistema Microkernel

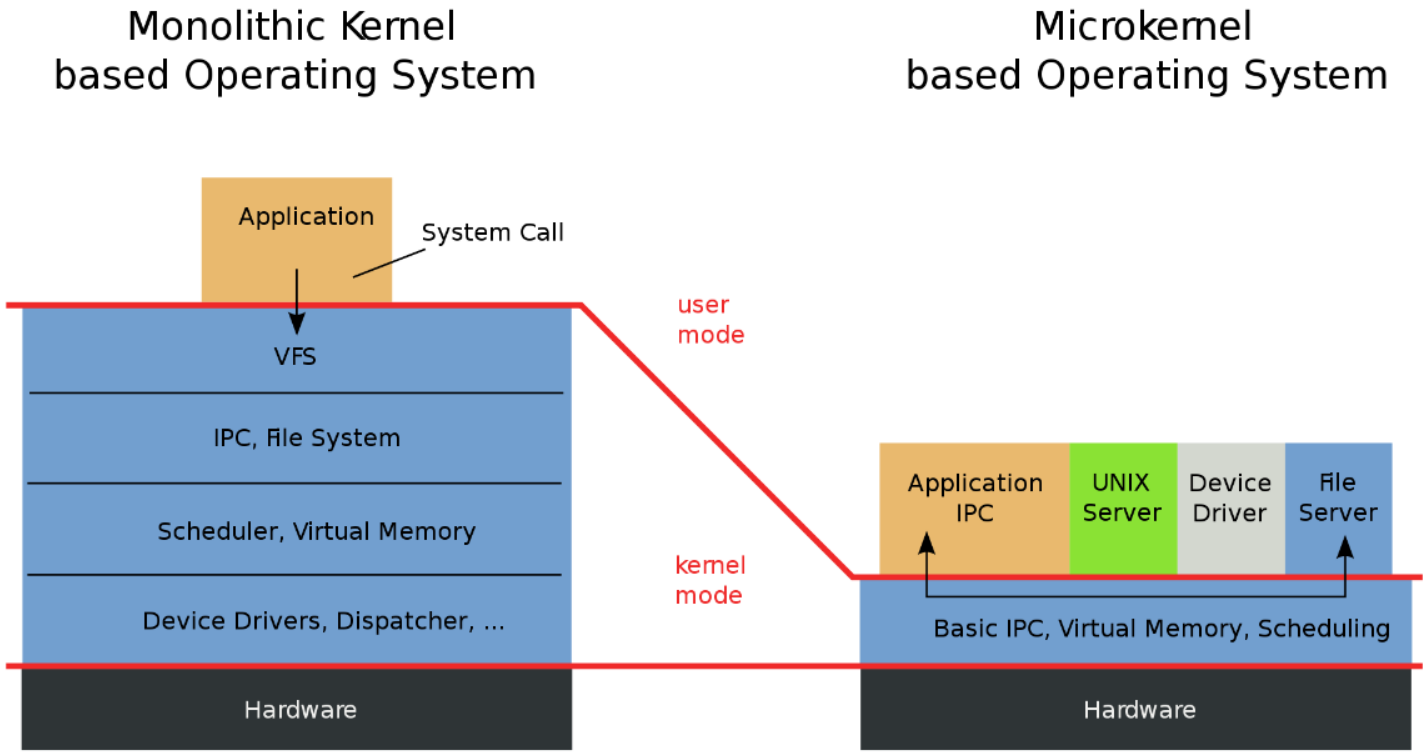
- Define un kernel muy simple (microkernel) que trabaja sobre el hardware. Solo las funciones absolutamente esenciales del SO deben permanecer en el microkernel.
- Los servicios y las aplicaciones menos esenciales se construyen sobre el microkernel y se ejecutan en modo usuario.
- El microkernel tiene un conjunto de primitivas y llamadas al sistema para implementar unos servicios mínimos del SO (comunicación entre procesos, gestión de memoria y planificación dela CPU)
- Los demás servicios que solían ser suministrados por el kernel, como por ejemplo el servicio de red, se implementan como programas de usuario conocidos como servidores.

## • Ventajas

- Añade más estabilidad al sistema por que un fallo de un servidor solamente pararía un programa simple, en lugar de provocar la parada del SO completo

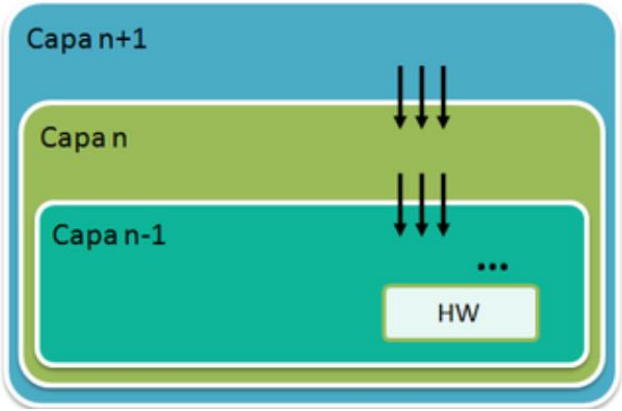
  Ejemplo: Minix, desarrollado por Andrew S.Tanembaum

# Sistema Monolítico vs  Sistema Microkernel



# Estructura de un SO: Sistema por capas

- La estructura del SO se divide en capas o niveles
- Jerarquía de capas donde cada una de ellas ofrece una interfaz clara y bien definida
- La capa superior solamente utiliza los servicios y funciones que ofrece la capa inferior
- El encargado de que solamente haya comunicación entre capas adyacentes es el procesador
- La capa más interna o inferior (capa 0) corresponde al hardware, mientras que la más alta o externa corresponde a la interfaz de usuario.




### Ventajas

- Al tener una organización por módulos, otorga facilidad en construcción y depuración del sistema.

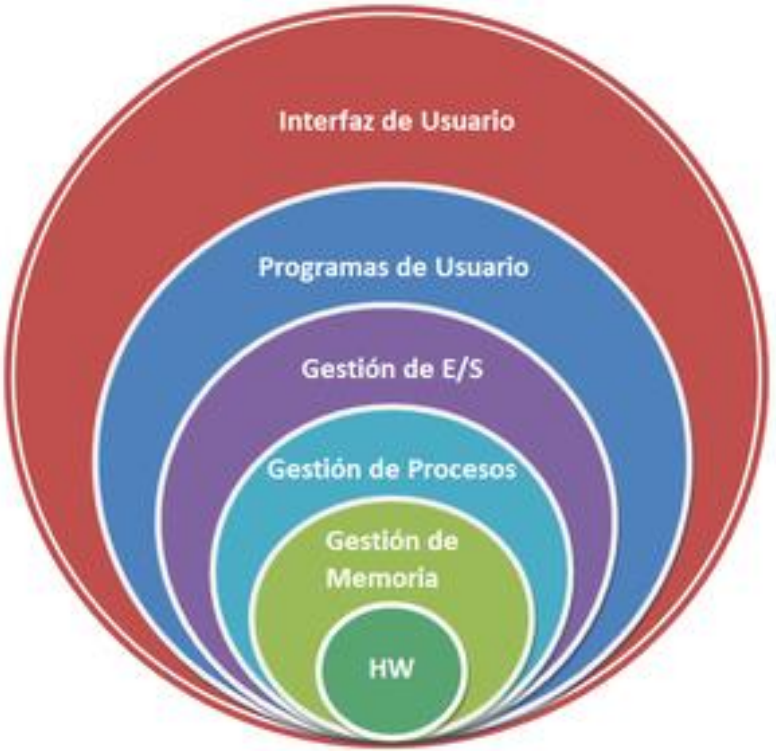
### Inconveniente

- Problemática al realizar la división y definición de las funcionalidades de cada capa ya que las capas superiores solo pueden utilizar servicios de la capa inferior

# Estructura de un SO: Sistema por capas

**El sistema original consta de 6 capas:**

- **Capa 5**: Se encuentra la interfaz de usuario.
- **Capa 4**: Aloja los programas de usuario
- **Capa 3**: Se controlan los dispositivos E/S
- **Capa 2**: Se administra la comunicación entre procesos y la consola del operador.
- **Capa 1**: Administración de memoria y discos.
- **Capa 0**: Correspondiente al hardware, realizando asignación del procesador, también alterna entre procesos cuando ocurren interrupciones o se han expirado y proporciona multiprogramación básica de la CPU.



# Estructura de un SO: Sistema por módulos

- El kernel se compone de módulos, y cada uno de estos módulos se encuentra separado de forma independiente, tal que, si alguno falla no afecta a los otros, ni al núcleo.
- Los módulos se pueden cargar dinámicamente en el núcleo cuando se necesiten, es decir, en tiempo de ejecución o durante el arranque del sistema
- El kernel dispone de los componentes fundamentales y se conectan directamente con servicios adicionales.
- Además otros componentes pueden cargarse dinámicamente al núcleo.
- En general, esta estructura se parece bastante a la de capas, pero es mucho más flexible debido a que cualquier módulo de esta estructura puede llamar a otro.
- La mayoría de los sistemas operativos modernos implementan este enfoque

  Ejemplo: Unix modernos, Solaris, Linux, Mac OSX

# Estructura de un SO: Máquina virtual

- Los SO de máquina virtual presentan a cada proceso una máquina que parece idéntica a la real
- Deben ser multitarea, con el objetivo de instalar sobre el mismo equipo diferentes sistemas operativos
- No es necesario que sean diferentes sistemas operativos, sino que pueden instalar varias réplicas de máquinas que tengan instaladas el mismo sistema operativo

# Estructura de un SO: Cliente - Servidor

- Los procesos del sistema operativo pueden ser tanto cliente como servidor
- Ej: Un programa de aplicación de un usuario que se está ejecutando se convertiría en un proceso cliente, que pide al sistema operativo servicios que serían los procesos servidores

# Índice


1 Sistema Operativo

2 Evolución histórica

3 Clasificación SO

4 Estructura de un SO

5 **Funciones de un Sistema Operativo**


# Funciones de un Sistema Operativo

- Un SO multiusuario proporciona un entorno dentro del cual se ejecutan las aplicaciones.
- Para construir este entorno se divide al SO lógicamente en pequeños módulos y se crea una interfaz bien definida entre estos módulos y las aplicaciones que se ejecutan.
- Cada uno de estos módulos se encarga de realizar una función específica.
- Entre estas están:
  - Administrador de procesos
  - Administrador de memoria principal
  - Administrador del almacenamiento secundario
  - Gestión de ficheros y directorios
  - Administrador de E/S
  - Protección

# Funciones de un Sistema Operativo: Administrador de procesos

- Un proceso es un programa en ejecución con su entorno asociado. Un programa puede generar varios procesos activos.
- Un proceso necesita determinados recurso como tiempo de la CPU, memoria, archivos y acceso a dispositivos E/S. Estos recursos se proporcionan al crear el proceso, o se le asignan mientras se ejecuta.
- Potencialmente, todos estos procesos pueden ejecutarse en forma concurrente, multiplexando la CPU entre ellos.
- El SO es responsable de las actividades relacionadas con la administración de procesos:
  - Crear y eliminar procesos.
  - Suspender y reanudar la ejecución de procesos.
  - Sincronización de procesos.
  - Comunicación entre procesos.
  - Análisis de interbloqueos
- Puede haber procesos del sistema y procesos de usuarios

# Funciones de un Sistema Operativo: Administrador de la memoria principal

- La memoria es el dispositivo de almacenamiento principal en un sistema informático.
- El procesador lee las instrucciones y lee o escribe datos de la memoria principal. Por otra parte, los dispositivos de E/S necesitan acceso a memoria para lectura y escritura de datos.
- Para mejorar la utilización de la CPU y la velocidad de respuesta del ordenador a los usuarios, se debe conservar varios programas en memoria.
- Las actividades relacionadas con la administración de memoria son las siguientes:
  - Llevar el control de las zonas de memoria usadas y quien las usa.
  - Decidir que procesos se cargan en memoria habiendo espacio libre.
  - Asignar y recuperar espacio de memoria.
- Dependiendo de si el sistema es monotarea o multitarea, las técnicas de gestión de memoria serán distintas.

# Funciones de un Sistema Operativo: Administrador del almacenamiento secundario

- Cuando la memoria principal sea pequeña para almacenar todos los programas y los datos, el sistema informático debe ofrecer almacenamiento secundario que la respalde, es lo que llamamos memoria secundaria.
- Este almacenamiento se hace generalmente mediante disco donde se sitúan habitualmente las aplicaciones que se cargan en memoria principal solo cuando se ejecutan.
- Respecto a la memoria secundaria las funciones van a ser:
  - Administración del espacio libre en dispositivos de almacenamiento.
  - Asignación del almacenamiento.
  - Planificación de las operaciones sobre el disco.
- Una de las operaciones principales del sistema será resolver las operaciones de carga y descarga de información desde el almacenamiento secundario a la memoria principal.

# Funciones de un Sistema Operativo: Gestión de ficheros y directorios

- Es una de las partes más visibles del SO por parte del usuario.
- Los ordenadores pueden almacenar información en varios dispositivos físicos, siendo los más comunes el disco duro, el disco óptico y las memorias flash.
- El SO ofrece una perspectiva lógica uniforme del almacenamiento de información, el archivo o fichero.
- Un archivo es un conjunto de información relacionada y que se almacena y manipula por el sistema operativo como una entidad única.
- Comúnmente los archivos contienen programas y datos. Los archivos se organizan en directorios para facilitar su acceso. Como varios usuarios tienen acceso a ellos, es deseable controlar quién tiene acceso a los archivos y cómo puede hacerlo.
- El SO es responsable de las siguientes actividades relacionadas con la administración de archivos:
  - Crear y eliminar archivos.
  - Crear y eliminar directorios.
  - Control de operaciones para manipular archivos y directorios.
  - Correspondencia entre archivos y almacenamiento secundario.
  - Copia de seguridad de archivos en medios de almacenamiento estables.

# Funciones de un Sistema Operativo: Gestión de E/S

- Un sistema informático tiene una diversidad de dispositivos periféricos conectados.
- Uno de los objetivos del SO es ocultar al usuario las particularidades que estos dispositivos hardware y facilitar su uso.
- Sus funciones básicas son:
  - Un sistema de memoria caché mediante buffer
  - Una interfaz general con los controladores de dispositivo y unos controladores para dispositivos hardware específicos

# Funciones de un Sistema Operativo: Protección

- La protección se refiere al control del acceso de las aplicaciones y usuarios a los recursos hardware de un sistema informático.
- El SO tiene la responsabilidad de proteger un proceso de los otros. Por tanto debe asegurar que los ficheros, segmentos de memoria, CPU y otros recursos de E/S puedan ser únicamente usados por aquellos procesos que hayan recibido la correspondiente autorización del sistema.
- Ejemplos:
  - El direccionamiento de memoria asegura que un proceso sólo puede trabajar dentro del espacio de direcciones asignado a ese proceso.
  - Hay que asegurar que ningún proceso pueda obtener el control de la CPU sin que lo acapare indefinidamente.
  - Por último, no se permite que los usuarios realicen por su cuenta sus operaciones de E/S, para proteger así la integridad de los dispositivos periféricos.

# Funciones de un Sistema Operativo: Interprete de comandos

Un **comando** (traducción literal del inglés command, «orden, instrucción») es una instrucción u orden que el usuario proporciona al SO, desde la línea de comandos (shell) o desde una llamada de programación.

Estos comandos pueden ser:

- Interno.- El código del comando está contenido en el propio archivo del intérprete.
- Externo.- Cuando el código del comando no está en el archivo del intérprete, sino en un archivo ejecutable aparte.

Suele admitir parámetros para modificar su comportamiento. Suelen indicarse tras una barra "/" (en SO Windows) o un guión simple "-" o doble "--" (en SO Unix/Linux)

Algunas de las consolas son:

- **command.com** para los sistemas basados en DOS (MS-DOS, PC-DOS, etc.). Ya está en desuso.
- **cmd.exe** para los sistemas basados en Windows con la aplicación Símbolo del sistema.
- **PowerShell** para Windows 8 y Server 2008 o superior.
- **bash, sh, csh, ksh,** etc. para los sistemas basados en Unix y GNU/Linux.

# Funciones de un Sistema Operativo: GUI o Interfaz Gráfica

- La interfaz gráfica de usuario, conocida también como GUI (Graphical User Interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.
- Su principal uso consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de un ordenador a los usuarios.
- Como ejemplos de interfaz gráfica de usuario, cabe citar los entornos de escritorio Windows, el X-Window de GNU/Linux o el de Mac OS X.
- Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con el ordenador. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no esta sujeto a los detalles de la visualización ni a la entrada a través del ratón o el teclado.
- Otra ventaja es que las aplicaciones escritas para una interfaz gráfica de usuario son independientes de los dispositivos