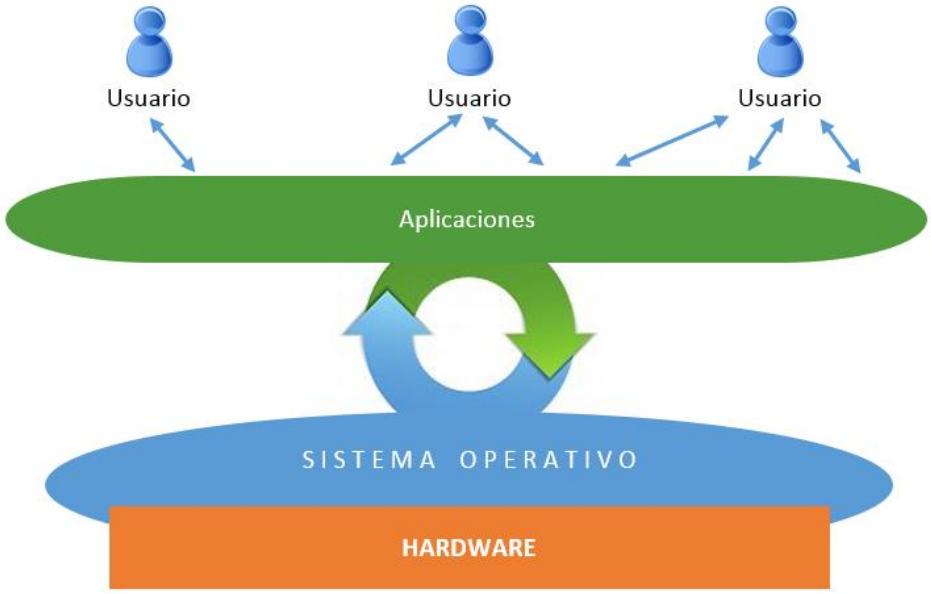
1. Fundamentos de sistemas operativos

1.1. Introducción

Un sistema operativo es el conjunto software que maneja tanto el hardware de un computador como los procesos que en él se ejecutan, a los cuales ofrece una serie de herramientas y servicios. De ello, podemos concluir que:

1. Las aplicaciones no forman parte del sistema operativo.
2. No cualquier sistema operativo puede ejecutarse en cualquier hardware.



En la actualidad existen diferentes tipos de sistemas operativos, orientados a cubrir diferentes necesidades:

- **Multitarea**. Permite la ejecución de diferentes tareas simultáneamente.
- **Multiproceso**. Soporta ejecutar programas en más de una CPU.
- **Multiusuario**. Permite que dos o más usuarios ejecuten programas simultáneamente.
- ***Multithread***. Soporta que diferentes partes de un mismo programa se ejecuten concurrentemente.
- ***Tiempo real***. Garantiza el procesado de eventos o datos en un tiempo específico.
- **Embebido / Móvil**. Diseñado para ejecutarse en pequeños dispositivos con autonomía y recursos limitados. Son compactos y hacen un uso del hardware extremadamente eficiente.
- **Distribuido**. Se ejecutan en diferentes computadores conectados por red, mostrándolos como uno único.

Así, por ejemplo, *Android Marshmallow* sería un sistema operativo móvil, multitarea, multiproceso, multiusuario y *multithread*.

1.2. Funciones de los sistemas operativos

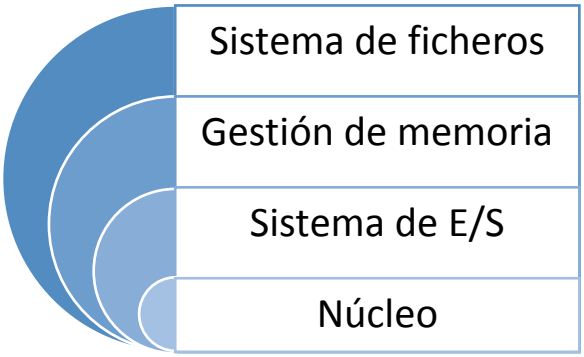
Un sistema operativo debe ofrecer las siguientes funcionalidades:

1. **Facilitar el desarrollo de programas.** Debe proporcionar herramientas (tales como editores, compiladores, depuradores, etc.) que faciliten la creación de nuevos programas.
2. **Ejecución de programas.** Ha de permitir la ejecución de programas de forma transparente para el usuario (carga del programa en memoria, acceso a periféricos…), así como facilitar la ejecución concurrente o paralela de los mismos.
3. **Acceso a periféricos (E/S).** Debe dar una visión unificada para acceder a los dispositivos que controla.
4. **Acceso controlado a ficheros.**
5. **Acceso al sistema.** En sistemas compartidos o públicos, debe controlar el acceso al sistema completo y sus dispositivos.
6. **Detección y gestión de errores.** Durante la ejecución se pueden producir errores, tanto hardware como software. El sistema operativo debe dar una respuesta adecuada cuando el error se produce, demostrando estabilidad y fiabilidad.
7. **Monitorización.** Colección de datos de rendimiento: tiempo de respuesta, tiempo de acceso, porcentaje de uso de los recurso, etc.

1.3. Arquitectura

Cualquier sistema operativo está formado por 4 módulos o subsistemas. El conjunto de estos cuatro bloques se conoce como *kernel*:

1. **Núcleo.** Proporciona el nivel más básico de control sobre el hardware del computador (con la ayuda de los controladores de dispositivos - *drivers*), gestiona (y si es necesario inhabilita) las interrupciones provocadas por el hardware, conmuta la CPU entre las diferentes tareas, decide qué proceso accede a dichos recursos hardware y ofrece los mecanismos y servicios básicos para el correcto desarrollo y ejecución de los procesos.
2. **Sistema de entrada/salida.** Gestiona cualquier acceso, tanto de entrada como de salida, a los periféricos. Cualquier bloque que quiera dialogar con el hardware tiene que comunicarse con esta capa.
3. **Gestión de memoria.** Maneja el uso de la memoria por parte de los procesos en ejecución concurrente (creación, indexación, destrucción…). Debe garantizar que un proceso no interfiere en la memoria de otro. Es el encargado de la gestión de la memoria virtual, en caso de haberla.
4. **Sistema de ficheros.** Provee una visión estructurada de los datos almacenados en el disco. El objetivo de esta capa es proporcionar un acceso rápido y fiable a la información, maximizando el uso de las capacidades del disco (o discos).

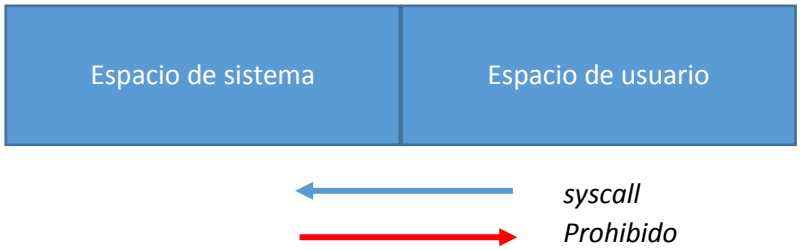


Además de estos bloques básicos, podemos encontrar otros, dependiendo de las características, funcionalidades y dimensiones del sistema:

5. **Gestión de redes.** Soporte de los diferentes protocolos de comunicación, del hardware de redes y de las aplicaciones que los usan. Computadores usando diferentes sistemas operativos han de ser capaces de comunicarse entre ellos.
6. **Seguridad.** Para que un computador sea seguro, el sistema operativo debe controlar a qué permite y a qué deniega acceso en el sistema. En sistemas multiusuario, métodos de autenticación son requeridos, así como el establecimiento de roles y niveles de privilegios.
7. **Interfaz de usuario.** Modo en el que el sistema va a mostrarse visualmente al usuario.

1.4. Espacios y tipos de kernels

Dentro de la memoria del computador tenemos el sistema operativo y los procesos (programas en ejecución). Para evitar que los procesos accedan a la memoria usada por el sistema operativo de manera incontrolada, el sistema segrega virtualmente (y a veces incluso físicamente) la memoria en espacio de sistema y espacio de usuario. Aquello que se ubica en el espacio de usuario puede acceder a aquello ubicado en el espacio de sistema mediante llamadas al sistema (*syscall*). Elementos en el espacio de sistema no pueden acceder a elementos en el espacio de usuario. Ello conlleva que la distribución en memoria sea un rompecabezas con diferentes ventajas y desventajas.



Dependiendo de donde se ubican los diferentes bloques, tendremos diferentes tipos de *kernel*:

- ***Kernel* monolítico.** La mayoría de servicios se encuentran en el espacio de sistema. Fáciles de desarrollar, se caracterizan por conllevar pocos cambios de contexto (alto rendimiento), pero los drivers (desarrollados por terceras partes) se instalan en el espacio

del sistema, lo cual supone una vulnerabilidad. Habitual en sistemas Linux. Un aspecto muy crítico es la portabilidad: suelen ser muy difíciles de portar a arquitecturas diferentes de aquellas para las que han sido diseñados, ya que son grandes porciones de código ligadas a un hardware en concreto. Aun así, podemos encontrar que muchos *kernels* han sido portados a diferentes arquitecturas mediante el uso de directivas de pre-compilación.

- ***Microkernel*.** El espacio de sistema contiene lo mínimo, lo más crítico (abstracción del hardware). Todo el resto se ubica en la memoria de usuario. Es mucho más seguro y estable, tiene un mejor tiempo de respuesta, pero el rendimiento es bajo debido a la enorme cantidad de cambios de contexto. Son mucho más fáciles de portar, ya que hay menos código a adaptar.
- ***Kernel* híbrido.** Son una mezcla de *kernel* monolítico y *microkernel*. Además de los servicios más críticos, se analiza el sistema para ubicar aquellos servicios del sistema más usados en el espacio de sistema, reduciendo así los cambios de contexto. Es el usado por la mayoría de sistemas operativos comerciales (Windows, Mac OS X).
- ***Exokernels*.** Enfoque completamente distinto. Reduce la gestión y abstracción del hardware, dando más responsabilidades y acceso más directo a las aplicaciones. Mantiene simplemente las tareas de protección y multiplexado de los recursos.
- ***Nanokernel*.** Delega casi todos los servicios, incluidos los de gestión de interrupciones, a los controladores de dispositivos (*drivers*), haciendo que el *kernel* casi no necesite memoria. Hay quienes lo consideran una variante de *exokernel* y no un tipo propio.