Estructura de un sistema operativo

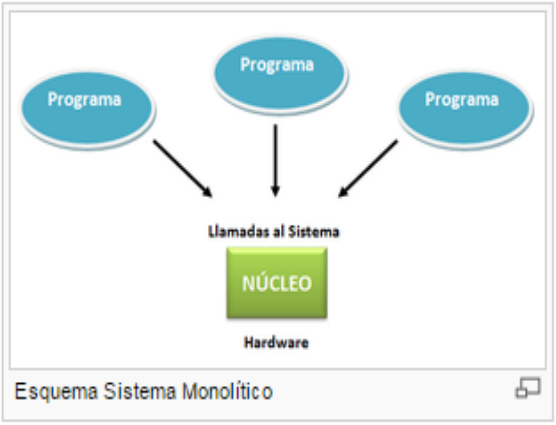
Analizando la historia de los Sistemas Operativos notamos que se puede considerar que éstos surgen desde finales de los 50's con una arquitectura bastante obsoleta comparada con la de la actualidad.
Para poder construir un Sistema Operativo se deben tener en cuenta dos tipos de requisitos, los cuales son:

- **Requisitos de usuario:** Un sistema fácil de usar y de aprender, seguro, rápido y adecuado para el uso que se le necesita dar.
- **Requisitos del software:** Considera el continuo mantenimiento, forma de operación, restricciones de uso, eficiencia, tolerancia frente a los errores y flexibilidad.

El objetivo de la estructuración es buscar una organización interna que facilite la comprensión, incremente la portabilidad, extensión y favorecer el mantenimiento de los Sistemas Operativos.

Los Sistemas Monolíticos. (Estructuras Simples)

Los sistemas Monolíticos son la estructura más simple para un Sistema Operativo. También llamados de Estructura Modular, fue escrito para proporcionar una máxima funcionalidad dentro del menor espacio posible. Se caracteriza porque no tienen una estructura totalmente clara, con esto nos referimos a que sus rutinas y funcionalidades (ej. manejo de *drivers*, sistemas de archivos, gestión de memoria, etc.), se encuentran agrupados en un solo programa (el Sistema Operativo).
Este sistema está descrito como un conjunto de procedimientos o rutinas entrelazadas de tal forma que cada una tiene la posibilidad de llamar a las otras rutinas cada vez que así lo requiera. Sin embargo, cabe destacar las falencias en este tipo de estructura que radica principalmente en la poca confiabilidad otorgada, ya que todo el sistema, al no tener una estructura definida, se ejecuta todo en el mismo nivel del núcleo (*kernel*) lo que lo hace altamente vulnerable, por esta razón cuando falla un programa se produce un error en todo el sistema.
Además, otro problema inherente al Sistema Monolítico es que si se modifica el hardware por lo general es necesario recompilar el *kernel* para poder disponer de las funcionalidades. Esto consume tiempo y recursos porque la compilación de un nuevo *kernel* puede durar varias horas y necesita de una gran cantidad de memoria. Cada vez que alguien añade una nueva característica o corrige un error, significa que se necesitará hacer una recompilación del *kernel* entero, un ejemplo de esto podemos verlo en Linux. También el hecho de que en el espacio del *kernel* están incluidos todos los servicios básicos, tiene tres grandes inconvenientes: *el tamaño del núcleo, la falta de extensibilidad y la mala capacidad de mantenimiento*.



Esquema Sistema Monolítico

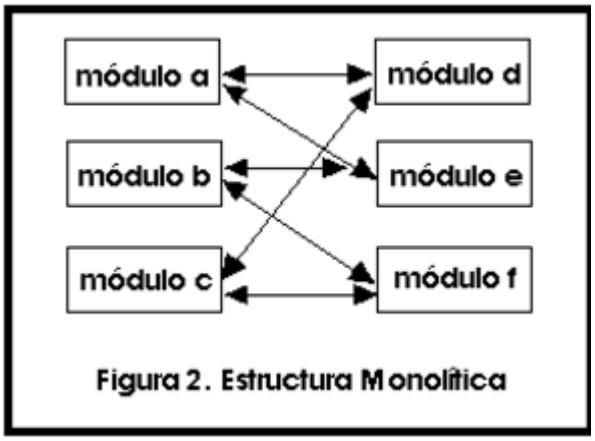
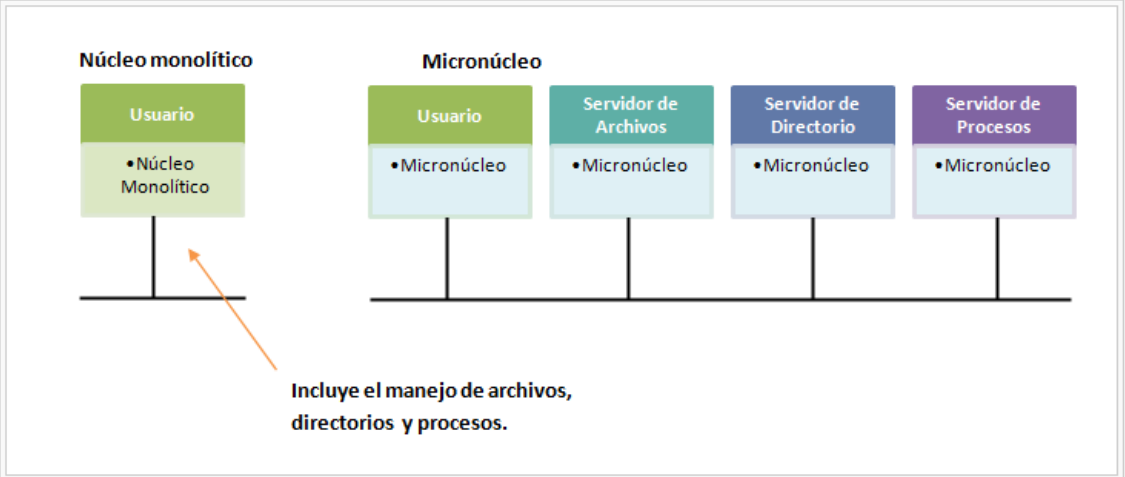


Figura 2. Estructura Monolítica

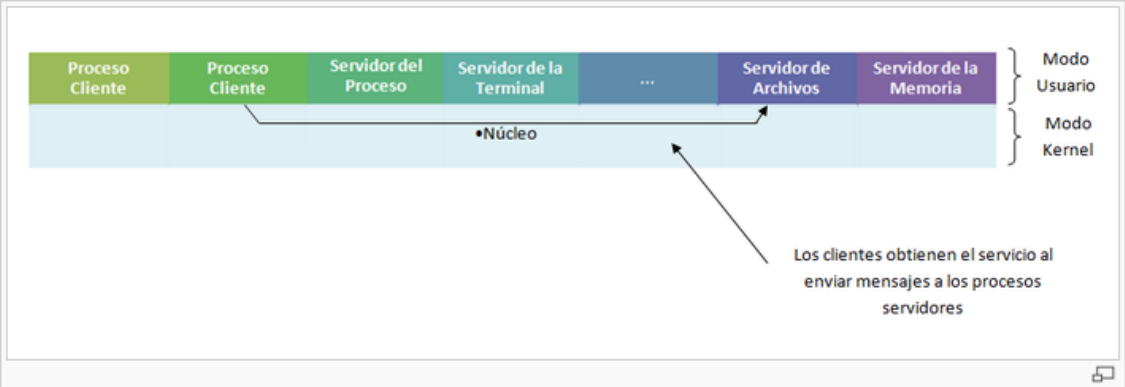
Los sistemas de Micronúcleo o Microkernel

El Micronúcleo surge como una nueva forma de organización para un Sistema Operativo, es un término algo tedioso de entender ya que puede no ser relativo a su tamaño, pero si a su diseño.
En este sistema las funciones centrales son manejadas por el núcleo (kernel) y la interfaz de usuario es manejada por el entorno (shell). El Microkernel se encarga de todo el código de un sistema, y de planificar los hilos (threads) con la finalidad de tener multitareas.
Algunas ventajas que podemos destacar de los Micronúcleos son los siguientes:
•**Uniformidad de interfaces:** disponen de una interfaz única para las solicitudes de los procesos, el paso de mensajes.
•**Portabilidad:** reduciendo el núcleo e implementando casi todo en servidores, para implementarlo en arquitecturas diferentes, sólo habría que modificar el núcleo haciendo más simple su portabilidad.
•**Fiabilidad:** es más fácil corregir fallas en un sistema pequeño ya que se pueden realizar pruebas más rigurosas que en un sistema mucho más grande.

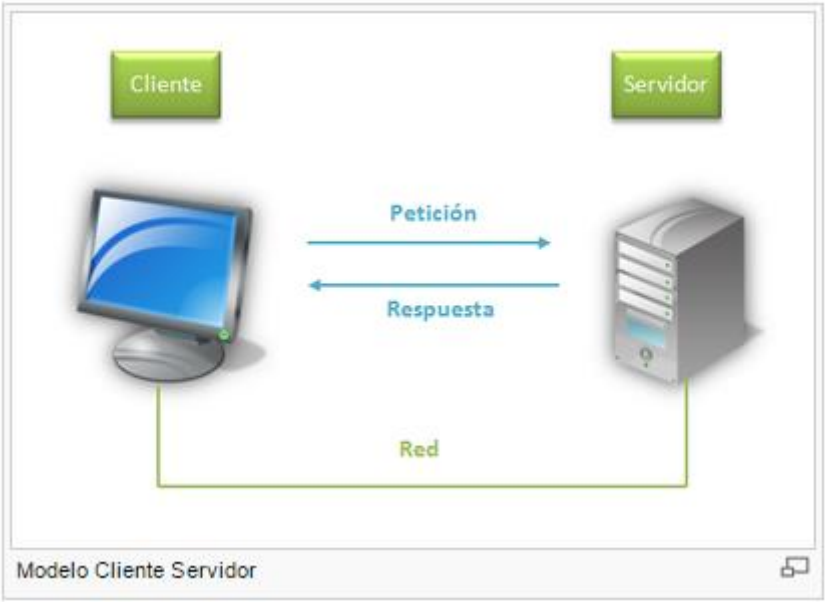


Esquema Sistema Monolítico-Micronúcleo

Sistema Cliente-Servidor:
Dentro de esta estructura también podríamos incluir el Sistema Cliente-Servidor ya que presenta una ligera variación en la idea del Microkernel la cual es que este sistema hace la diferencia entre dos clases de procesos: los servidores, cada uno de los cuales proporciona cierto servicio, y los clientes, que utilizan estos servicios. A menudo la capa inferior es un microkernel, pero eso no es requerido. La esencia es la presencia de procesos cliente y procesos servidor.
En los sistemas operativos modernos, los sistemas cliente-servidor nacen con la finalidad de minimizar el núcleo (kernel), trasladando el código de todos sus servicios a las capas superiores; y el núcleo sólo deberá controlar la comunicación, que se realiza mediante mensajes, entre clientes y servidores o servidores y hardware.
El objetivo es desarrollar la mayoría de las funciones del sistema operativos como procesos de usuario. Un proceso de usuario, llamado en este caso proceso cliente, envía una solicitud a un proceso servidor, que realiza el trabajo y devuelve la respuesta.
El sistema operativo se divide en partes donde cada una controla una faceta del sistema, entre ellos servicios a archivos, servicios a procesos, servicios a terminales, o servicios a la memoria, donde cada una es pequeña y controlable, así al ejecutar los procesos en modo usuario y no en modo núcleo si hay algún error en algún servidor, este afectará sólo a dicha parte y no a toda la máquina, ya que no se tiene acceso al hardware.



Un caso sencillo de cliente, en este caso, es un programa de aplicación que llama al servidor para acceder a un archivo, otro ejemplo es cuando el programa de aplicación realiza una operación de entrada o salida a un dispositivo concreto. En cada uno de estos casos el cliente a su vez puede ser servidor de otros servicios. Esta idea se refleja a continuación:



Los sistemas por capas o jerárquica (Estructura por niveles)

En esta estructura el Sistema Operativo queda definido modularmente por divisiones en capas o niveles, cuya organización está dada como una 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, es decir, la capa *n* sólo se comunica para obtener lo requerido con la capa *n-1* (*Ver imagen debajo*), donde la capa inferior es la más privilegiada. 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.

El primer sistema construido de esta manera fue el sistema THE (*Technische Hogeschool Eindhoven*), desarrollado en Holanda por E. W. Dijkstra (1968) y sus estudiantes.

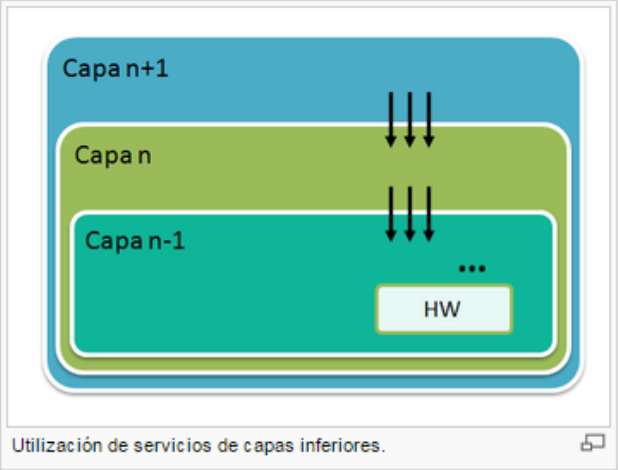
El sistema original consta de 6 capas:

Capa 5: Se encuentra la interfaz de usuario.

Capa 4: Aloja los programas de usuario.

Capa 3: Se controlan los dispositivos E/S (entrada y salida).

Capa 2: Se administra la comunicación inter-proceso 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.



Utilización de servicios de capas inferiores.

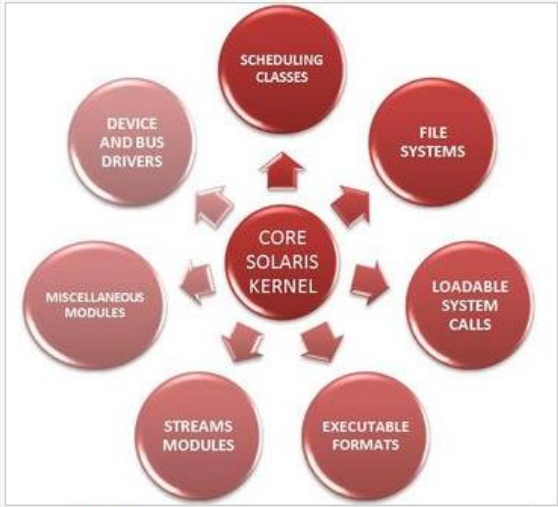
Como ventajas de este sistema podemos mencionar que al tener una organización modularizada, otorga facilidad en construcción y depuración del sistema. La facilidad de construcción se respalda porque al existir esta división en módulos (capas) se produce una abstracción del problema, simplificándose solamente a la función que realiza el módulo correspondiente a una capa N. También al lograr esta abstracción, no es necesario saber detalles de implementación de las capas inferiores, sólo se utilizan. La facilidad de depuración, quiere decir, que sea más simple la tarea de encontrar errores en el código y corregirlos. Otro aspecto positivo relacionado con la modularidad existente, cuando ocurre un error o falla en una de las capas, no se compromete a todo el sistema, sólo a la capa relacionada con la falla.
Con respecto a las desventajas de esta organización, al realizar la construcción de las capas, la problemática es la forma de realizar la división y definición de las funcionalidades, ya que se tiene considerar que las capas superiores solamente pueden utilizar los servicios de la capa que se encuentra inferior, por lo tanto, se debe tener mucho cuidado en la planificación del sistema para que exista un óptimo funcionamiento. Otra desventaja que podemos mencionar es el gasto de tiempo que se genera en ir de una capa a otra, cada capa implica un gasto extra.

Organización de Sistema por capas

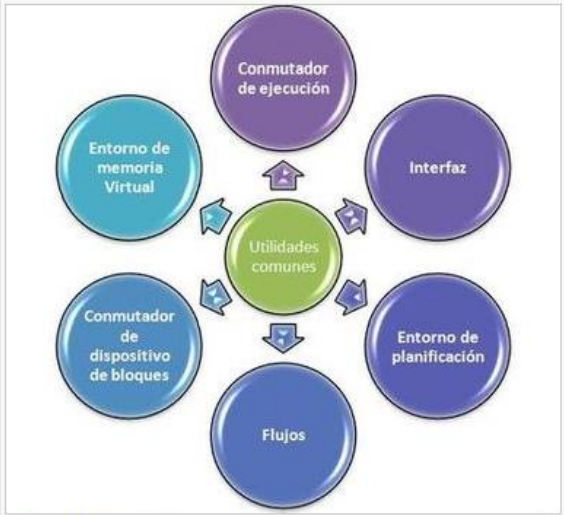
Sistemas por módulos

La mayoría de los sistemas operativos modernos implementan este enfoque. Lo que caracteriza este tipo de estructura es que el kernel se compone por 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, por ejemplo, si el módulo de software que se encarga de controlar el proceso de Telnet en una unidad se bloquea o es atacado, sólo este proceso se verá afectado. El resto de las operaciones siguen sus funciones habituales. Los módulos se pueden cargar dinámicamente en el núcleo cuando se necesiten, ya sea, 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. Este enfoque modular utiliza la programación orientada a objetos.

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. Es similar a la estructura de microkernel, pues el kernel también tiene las funciones esenciales, pero este es más eficiente ya que, no necesitan un mecanismo de paso de mensajes para comunicarse, sólo interfaces conocidas.



Esquema Módulo Solaris



Núcleo Unix Moderno