Introducción a los sistemas operativos

Definición de Sistema Operativo

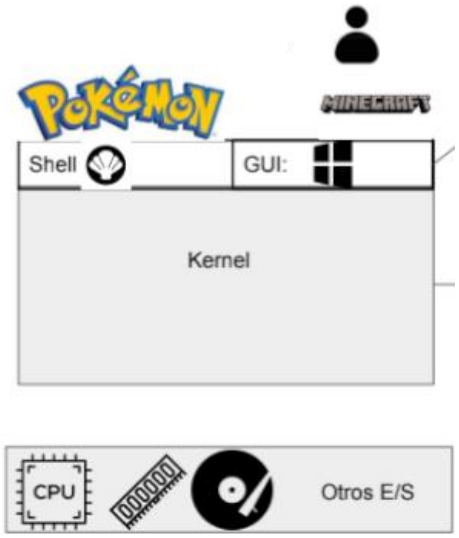
Hagamos una mínima introducción, de: ¿qué hablamos cuando hablamos de “Computadoras”? Una computadora **consta de:**

- Uno o más procesadores
- Una memoria principal
- Discos
- Otros dispositivos de E/S

Las computadoras están equipadas con una capa de software llamada sistema operativo, cuyo trabajo es **proporcionar a los programas de usuario un modelo mejor, más simple y pulcro, así como encargarse de la administración de todos los recursos antes mencionados**.

El Sistema Operativo proporciona a los **programadores** de aplicaciones y **programas** de aplicaciones un **conjunto abstracto de recursos simples**, en vez de los complejos conjuntos de hardware; y **administra estos recursos de hardware** de forma eficiente **siendo el intermediario entre el usuario y dicho hardware**.

Vamos a verlo más gráficamente



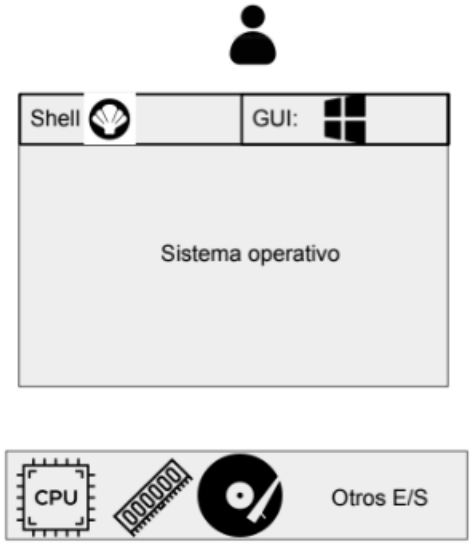
El programa más básico con el que los usuarios generalmente interactúan en entornos GNU/Linux de cloud y servidores se denomina shell, cuando está basado en texto, y GUI (Graphical User Interface; Interfaz gráfica de usuario) cuando utiliza elementos gráficos o íconos y, en sistemas Unix, **no forma parte del sistema operativo, aunque lo utiliza para llevar a cabo su trabajo.**

- En la parte inferior se muestra el hardware.
- Por encima del hardware se encuentra el **Sistema Operativo que no deja de ser un software** que **puede trabajar en dos modos: modo kernel** que tiene **acceso completo a todo el hardware** y puede ejecutar cualquier instrucción que la máquina sea capaz de ejecutar

Y **el resto del software se ejecuta en modo usuario donde un subconjunto de las instrucciones de máquina es permitido.**

Esto no se debe confundir con la ejecución a nivel usuario root. Los comandos ejecutados por root **se ejecutan dentro del modo de usuario, el modo kernel** hace referencia a las instrucciones privilegiadas ejecutadas **por máquina.**

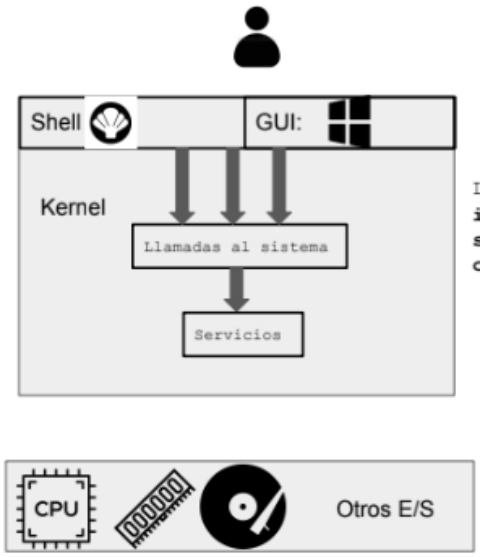
Básicamente, lo que es cualquier instrucción que suene a privilegiado, lo van a tener que hacer a través del Sistema Operativo, más específicamente el kernel.



El **sistema operativo está presente para administrar todas las piezas de un sistema complejo** y a**signa ordenadamente estas piezas como por ejemplo los accesos a todos los dispositivos de salida prácticamente**: pantallas, impresora, o escritura en disco, **inclusive al procesador: no acceden todos al mismo tiempo.**

El **kernel es definido como el núcleo o corazón del sistema operativo**, y se encarga principalmente de **mediar entre los procesos de usuario y el hardware disponible en la máquina** conteniendo las operaciones más que fundamentales.

Llamadas al sistema



Pero, hay una cosa más que tenemos que ver abajo del motor. Las llamadas al sistema.
Estas **proporcionan una interfaz con la que poder invocar los servicios que el sistema operativo ofrece**, como por ejemplo Entrada salida, protección o detección de errores.

Dicho en fácil… si un proceso está ejecutándose en modo usuario y necesita por ejemplo leer disco, solo puede hacerlo ejecutando la **llamada al sistema correspondiente.**

Normalmente, los desarrolladores de aplicaciones diseñan sus programas utilizando una API para acceder a esta **"interfaz con la que poder invocar los servicios que el sistema operativo ofrece"** . La API especifica un conjunto de funciones que el programador de aplicaciones puede usar, indicándose los parámetros que hay que pasar a cada función y los valores de retorno que el programador debe esperar.

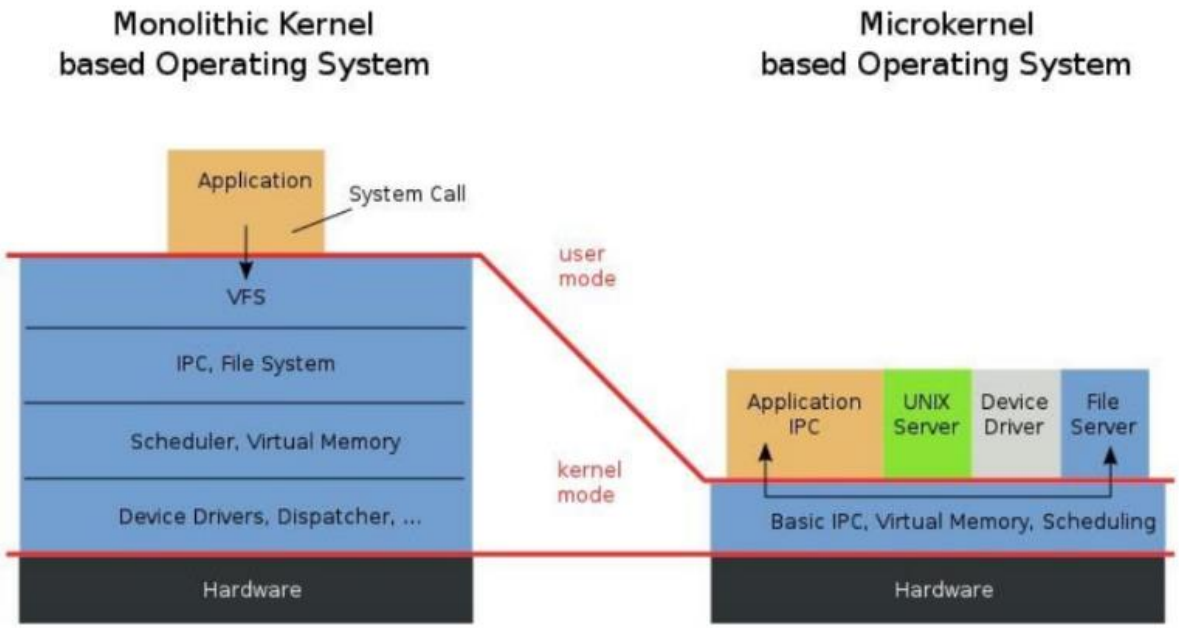
Las API disponibles para program adores son:

- API Win32 para sistem as W indows,
- La API POSIX para sistemas basados en UNIX

Una ventaja de programar usando una API está relacionada con la portabilidad: un programador de aplicaciones diseña un programa usando una API cuando quiere poder compilar y ejecutar su programa en cualquier sistema que soporte la misma API.

O sea, mientras mi sistema Operativo sea **POSIX compatible**, voy a poder implementar la librería que invoca las llamadas al sistema de la misma manera.

Tipos de Kernel



Hablamos de Kernel, hablamos que era el corazón del sistema operativo. Bien… pero estos Kernel pueden ser de varios tipos, Pero hay dos que nos importan:

Kernel monolítico

En este diseño, que hasta ahora se considera como la organización más común, **todo el sistema operativo se ejecuta como un solo programa en modo kernel** en un solo programa binario ejecutable extenso.
Para solicitar los servicios (llamadas al sistema) que proporciona el sistema operativo, los parámetros se colocan en un lugar bien definido (por ejemplo, en la pila) y luego se ejecuta una instrucción de trap que cambia la máquina del modo usuario al modo kernel

Muchos sistemas operativos soportan extensiones que se pueden cargar, como los drivers de dispositivos de E/S y sistemas de archivos. Estos componentes se cargan por demanda.

MIcrokérnel

Este método estructura el sistema operativo eliminando todos los componentes no esenciales del kemel e implementándolos como programas del sistema y de nivel de usuario; el resultado es un kemel más pequeño.

No hay consenso en lo que se refiere a qué servicios deberían permanecer en el kemel y cuáles deberían implementarse en el espacio de usuario.
Normalmente los microkernels proporcionan una gestión de la memoria y de los procesos mínima, además de un mecanismo de comunicaciones; pero un cambio de contexto significativamente mayor