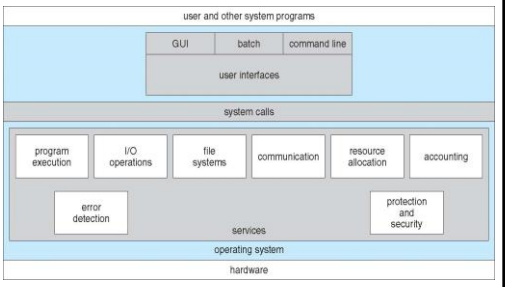
Estructura del Sistema Operativo

Estructuras de Sistemas Operativos

- Servicios de Sistemas operativos
- Interfaz de Usuario del Sistema Operativo
- Llamadas a Sistema
- Tipos de Llamadas a Sistema
- Programas de Sistemas
- Diseño e Implementación de un Sistema Operativo
- Estructura de un Sistema Operativo
- Generación y Boot del Sistema
- Conceptos de Máquinas Virtuales

Servicios del Sistema Operativo

- Un conjunto de servicios del SO proveen funciones que son útiles al usuario:
  - Interfaz de Usuario
  - Ejecución de Programas
  - Operaciones de E/S
  - Manipulación del Sistema de Archivos
  - Comunicaciones
  - Detección de errores
  - Y otros: alocación de recursos, contabilidad, protección ..



Interfaz de Usuario del Sistema Operativo

1.- Interfaz de líneas de comando (Command Line Interface - CLI) o *intérprete de comando* permite entrar comandos en forma directa, pueden ser por línea de comandos o gráficas:

- Algunas veces implementadas en el kernel, otras como programas de sistema
- La implementación a veces está embebida, y en otras es invocación a programas.

2.- Interfaz Gráfica (GUI)

3.- Interfaz Touch (especialmente en móviles)

Interfaz de Usuario del Sistema Operativo - GUI

Solaris – CDE (Common Desktop Environment)

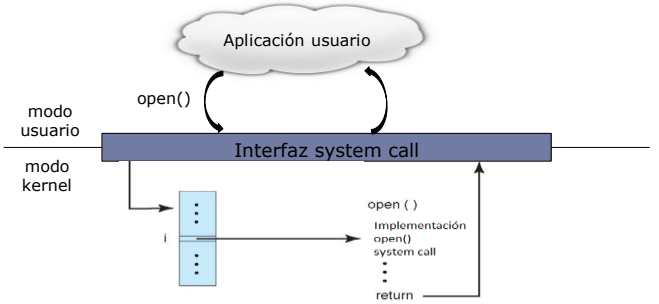
Mac OS GUI

Android

iOS

Llamadas al Sistema

- Son la interfaz de programación a los servicios provistos por el SO
- Típicamente escritas en lenguajes de alto nivel (C o C++)
- Mayoritariamente accedidas por programas vía *Application Program Interface (API)* más que por el uso llamadas a sistema directas



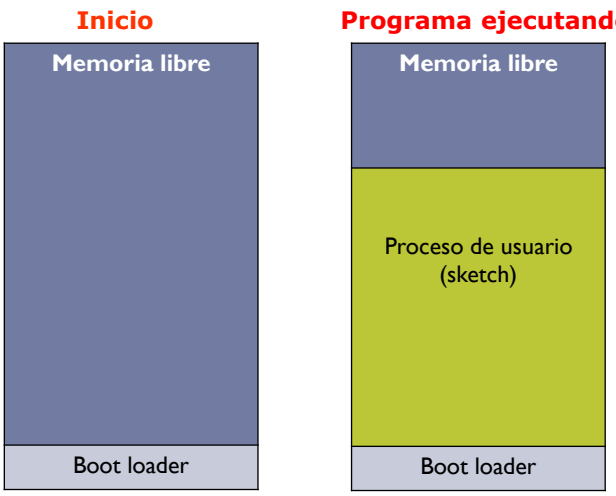
Pasaje de Parámetros en Llamadas a Sistema

- Métodos para pasar parámetros al SO
  - Parámetros en *registros*
  - Parámetros almacenados en un *bloque,* o tabla, en memoria, y la dirección del bloque pasada como parámetro en un registro.
  - Parámetros ubicados , o *pushed,* en un *stack* por el programa y *popped* del stack por el SO.

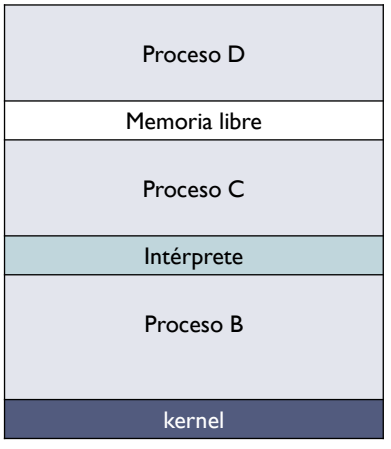
Tipos Llamadas a Sistema

- Control de procesos
  - create process, terminate process
  - end, abort
  - ...
- Administración de archivos
  - create file, delete file
  - open, close file
  - ...
- Administración de dispositivos
  - request device, release device
  - read, write, reposition
  - ...
- Mantenimiento de Información
  - get time or date, set time or date
  - get system data, set system data
  - …
- Comunicaciones
  - create, delete communication connection
  - send, receive messages
- Protección

Una tarea: ejemplo de ejecución en Arduino



Múltiples Tareas: ejemplo ejecución en FreeBSD



Programas de Sistema

- Los programas de sistema proveen un medio conveniente para el desarrollo de programas y ejecución. Pueden ser divididos en:
  - Manipulación de archivos
  - Información de estado
  - Modificación de archivos
  - Soporte de lenguajes de programación
  - Carga de programas y ejecución
  - Comunicaciones
  - Programas de aplicación

  La visión que tienen la mayoría de los usuarios del sistema operativo está dada por los programas de sistema y no por las llamadas a sistema (system calls).

Diseño e Implementación de un Sistema Operativo

- Los objetivos y las especificaciones están influenciados por la elección del hardware, tipo de sistema
- Objetivos de los *Usuarios* y los objetivos del *Sistema*
  - Objetivos de los Usuarios – El SO debe ser conveniente para su uso, fácil de aprender, confiable, seguro y rápido
  - Objetivos del Sistema – El SO debería ser fácil de diseñar, implementar y mantener, también flexible, confiable, libre de errores y eficiente

Asociado con los puntos de vista de un SO

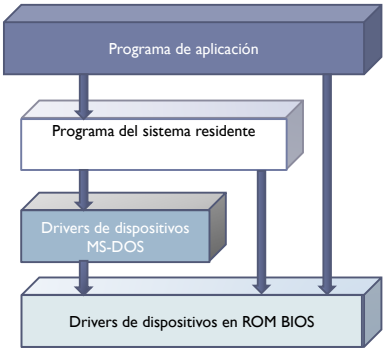
Diseño e Implementación de un Sistema Operativo

- Importante principio de separación

  Política: ¿Qué deberá hacerse?
  Mecanismo: ¿Cómo hacerlo?
- Los mecanismos determinan como hacer algo, las políticias deciden que debe hacerse
  - La separación de política de mecanismo es un principio muy importante, permite máxima flexibilidad si las decisiones políticas son cambiadas más tarde

Estructura Simple – MS-DOS

- **CASO MS-DOS**
  - Escrito para proveer máxima funcionalidad en el menor espacio
  - No está dividido en módulos
  - Aunque MS-DOS tiene cierta estructura, sus interfaces y niveles de functionalidad no están bien separados

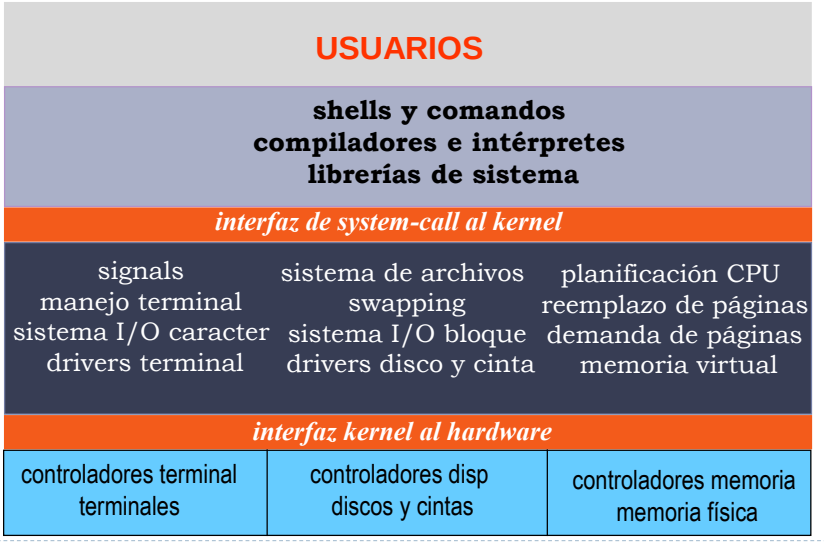


Estructura Simple - UNIX

- **CASO UNIX**
  - Está limitado por la funcionalidad del hardware, el sistema operativo UNIX original tenía una estructura limitada.

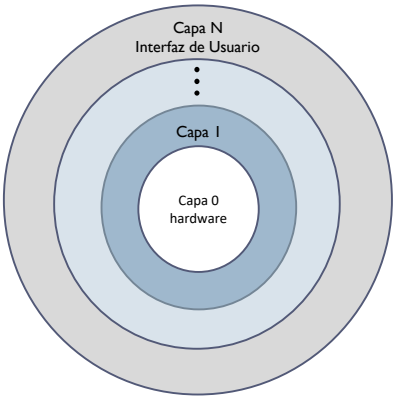
  El SO UNIX consiste de dos partes separables.
  - Programas de sistema
  - El kernel
    - Consiste de todo lo que esta debajo de la interfaz de los system calls y encima del hardware
    - Contiene el sistema de archivos, la planificación de CPU, manejo de memoria, y otras funciones del sistema operativo; un gran número de funciones en un solo nivel.

Estructura Simple - UNIX



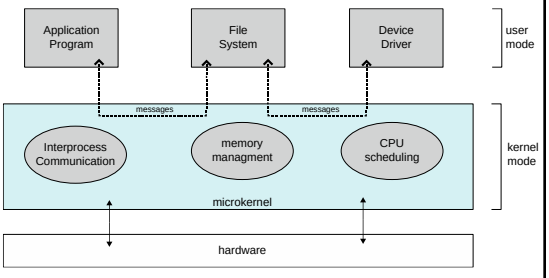
Enfoque por Capas

- El sistema operativo está dividido en un número de capas (niveles), cada una construída sobre el tope de otra. La capa inferior (nivel 0), es el hardware; la más alta (capa N) es la interfaz de usuario.
- En forma modular, las capas son seleccionadas de manera que cada una usa funciones (operaciones) y servicios de las capas inferiores.

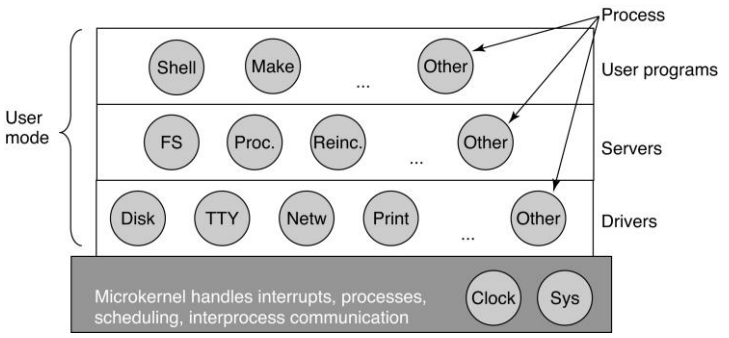


Estructura de Sistema Microkernel

- Mueve tanto como se pueda al espacio de *usuario*
- Las comunicaciones entre módulos de usuarios se realiza por medio de pasajes de mensajes
- Beneficios:
  - Más confiable (menos código corre en el modo kernel)
  - Más fácil de portar el SO a nuevas arquitecturas
  - Más fácil de extender
  - Más seguro
- Detrimentos:
  - Sobrecarga de rendimiento en la comunicación del espacio de usuario al espacio de kernel

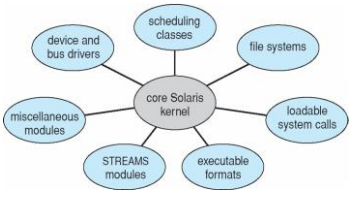


Sistema Microkernel – Ejemplo: Minix 3



Sistemas Modulados

- Los más modernos SOs implementan el kernel en módulos
  - Usa un enfoque orientado a objetos
  - Cada componente del núcleo está separado
  - Los protocolos de comunicación entre ellos son sobre interfaces conocidas
  - Cada uno es cargado en la medida que sea necesitado dentro del kernel
- En resumen, similar a capas pero más flexible
- Un ejemplo es Solaris



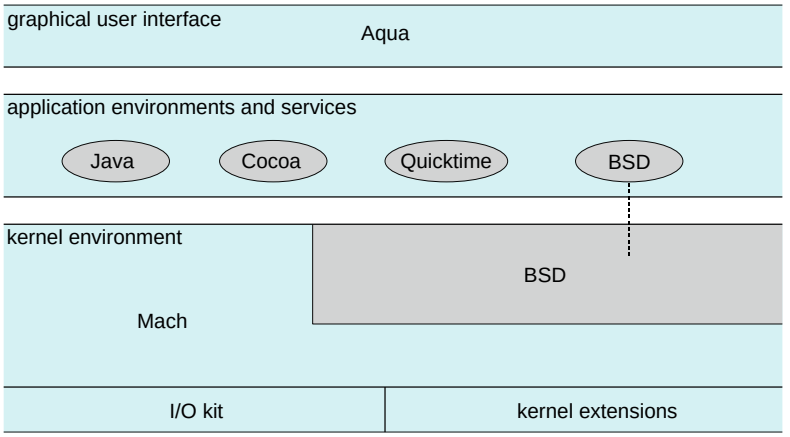
Sistemas Híbridos

Los sistemas operativos modernos no presentan un modelo puro.

Los modelos híbridos combinan multiples aproximaciones para alcanzar rendimiento, seguridad, usabilidad.

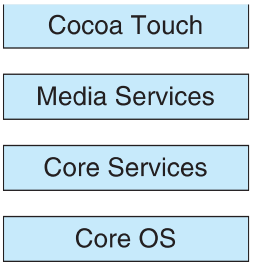
- Kernels de Linux y Solaris: en el espacio de direcciones del kernel presentan características monolíticas, además modulación para la carga dinámica de funcionalidades.
- Windows en su mayoría monolítico, además microkernel para diferentes subsistemas.
- Apple Mac OS X híbrido, por capas, **Aqua** UI más el ambiente de programación **Cocoa.**
  Kernel formado por un microkernel Mach y partes de BSD Unix, más un kit de E/S y la carga dinámica de módulos (llamados extensiones del kernel)

Sistemas Híbridos - Estructura de Mac OS X



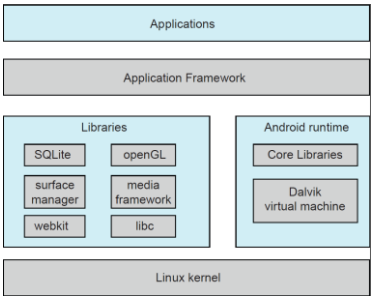
Sistemas Híbridos - iOS

- SO de Apple móvil para ***iPhone***, ***iPad***
  - Estructurado sobre Mac OS X, agregando funcionalidades para móviles.
  - No ejecuta directamente aplicaciones Mac OS.
  - **Cocoa Touch** Objective-C API para desarrollo de aplicaciones.
  - **Media services** capa para gráficos, audio y video.
  - **Core services** prove cloud computing, bases de datos.
  - Core operating system, basado en el kernel del Mac OS X.



Sistemas Híbridos - Android

- Basado sobre un kernel Linux kernel con modificaciones
  - Provee soporte para procesos, memoria, manejadores de dispositivos. Agrega administración de la energía
- Runtime incluye librería para el conjunto del núcleo y la máquina virtual Dalvik.
- Librerías incluyen frameworks para web browser (webkit), base de datos (SQLite), multimedia, pequeño libc.



Generación y Boot del Sistema

- Los sistemas operativos son diseñados para ejecutar sobre diferentes clases de computadora. El sistema debe configurarse para cada computadora específica.
- Programa SYSGEN obtiene información sobre la especificación de hardware al momento de configurar el sistema.
- El SO debe estar disponible al hardware, entonces el hardware puede iniciarlo
  - Pequeñas piezas de código – **bootstrap loader**, localiza el kernel, lo carga en memoria, y lo pone en marcha
  - A veces es un proceso en dos pasos donde el **boot block** en una locación fija carga el bootstrap loader
  - Cuando se le da energía y se inicializa el sistema, comienza la ejecución a partir de una dirección fija de memoria
    - Firmware es usado para contener el código inicial de boot

Conceptos de Máquinas Virtuales

- Una *máquina virtual* lleva la propuesta por capas a su conclusión lógica. Trata el hardware y el kernel del sistema operativo como si fuera todo hardware.
- Una máquina virtual provee una interfaz *idéntica* al hardware primitivo subyacente.
- El sistema operativo crea la ilusión de múltiples procesos, cada uno ejecutando en su propio procesador con su propia memoria (virtual).
- Cada invitado es provisto con una copia (virtual) de la computadora.

Bibliografía:

- Silberschatz, A., Gagne G., y Galvin, P.B.; "*Operating System Concepts*", 7ma Edición 2009, 9na Edición 2012, 10ma Edición.
- Tanenbaum, A.; "*Modern Operating Systems*", Addison-Wesley, 3ra Edición 2008, 4ta Edición 2014.