ESTRUCTURA DEL SISTEMA OPERATIVO

DEPARTAMENTO DE CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN

UNIVERSIDAD NACIONAL DEL SUR

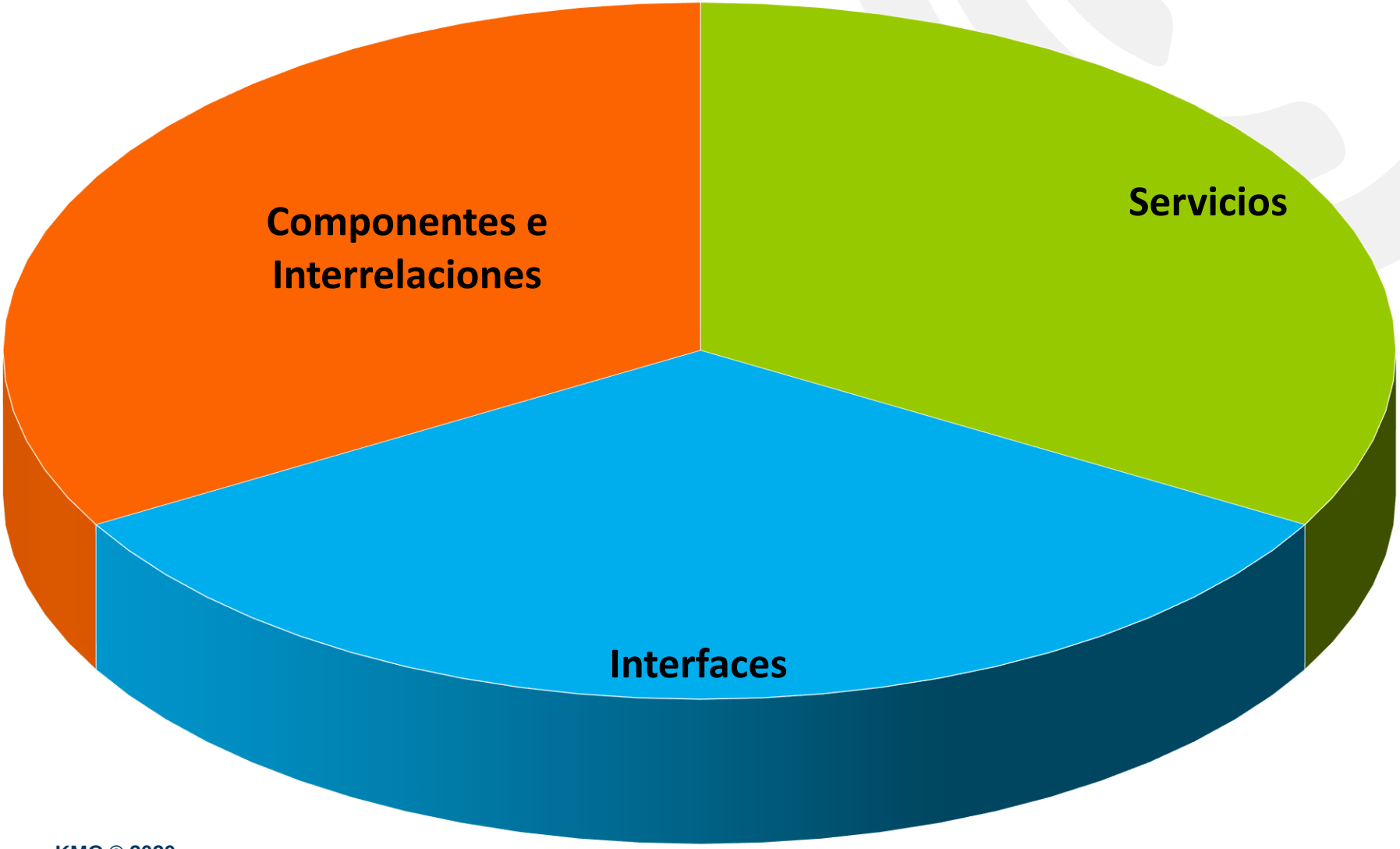
AGENDA

1. Conceptos
    1. Servicios de Sistemas operativos
    2. Interfaz de Usuario del Sistema Operativo
2. Llamadas a Sistema
    1. Tipos de Llamadas a Sistema
    2. Pasaje de Parámetros
3. Programas de Sistemas
4. Diseño, Implementación y Estructura de un Sistema Operativo
5. Generación y Boot del Sistema
6. Conceptos de Máquinas Virtuales

AGENDA

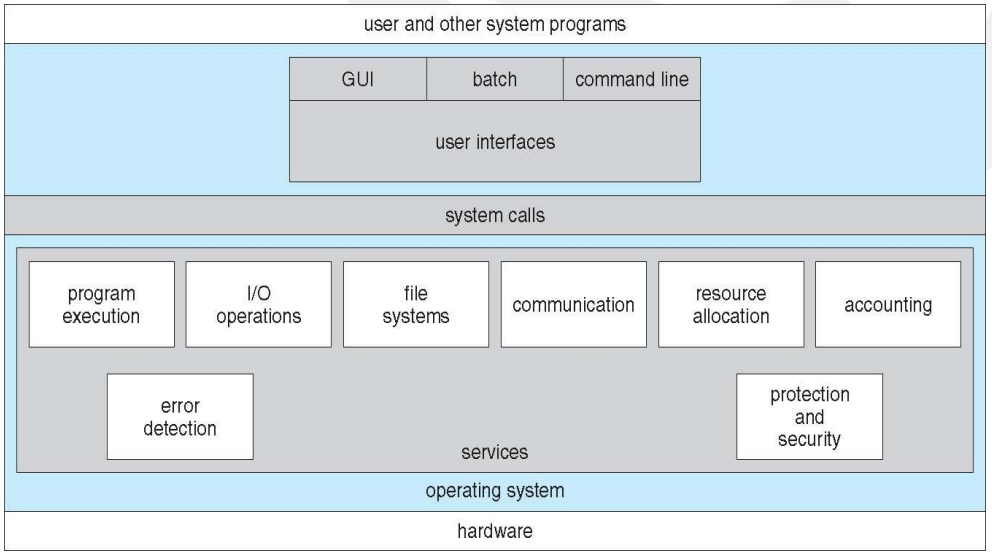
1. **Conceptos**
    1. **Servicios de Sistemas operativos**
    2. **Interfaz de Usuario del Sistema Operativo**
2. Llamadas a Sistema
    1. Tipos de Llamadas a Sistema
    2. Pasaje de Parámetros
3. Programas de Sistemas
4. Diseño, Implementación y Estructura de un Sistema Operativo
5. Generación y Boot del Sistema
6. Conceptos de Máquinas Virtuales

VISTAS DE UN SISTEMA OPERATIVO



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 errors
  - 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:

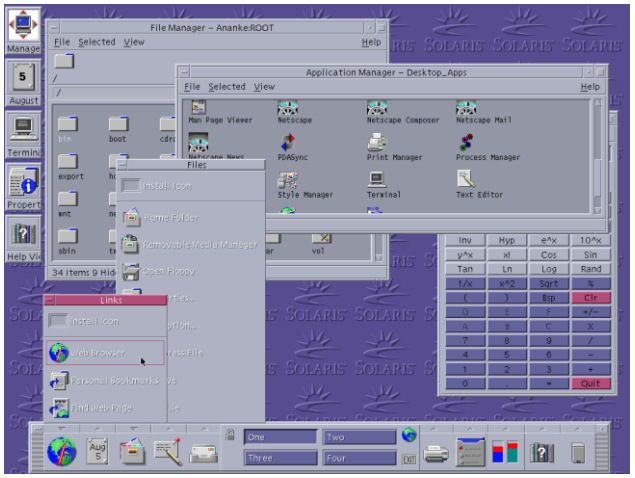
- Algunas veces implementadas en el kernel, otras como programas de sistema
- La implementación a veces está embebida, y en otras es invocación a programas.

2.- Interfaz Gráfica (GUI)

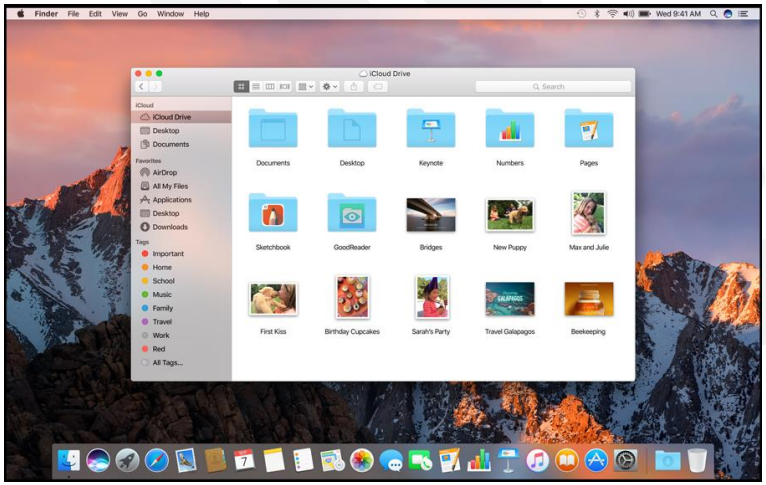
3.- Interfaz Touch (especialmente en móviles)

INTERFAZ DE USUARIO DEL SISTEMA OPERATIVO - GUI

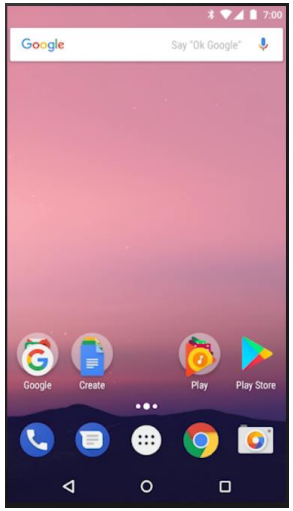
Solaris – CDE (Common Desktop Environment)



Mac OS GUI



Android



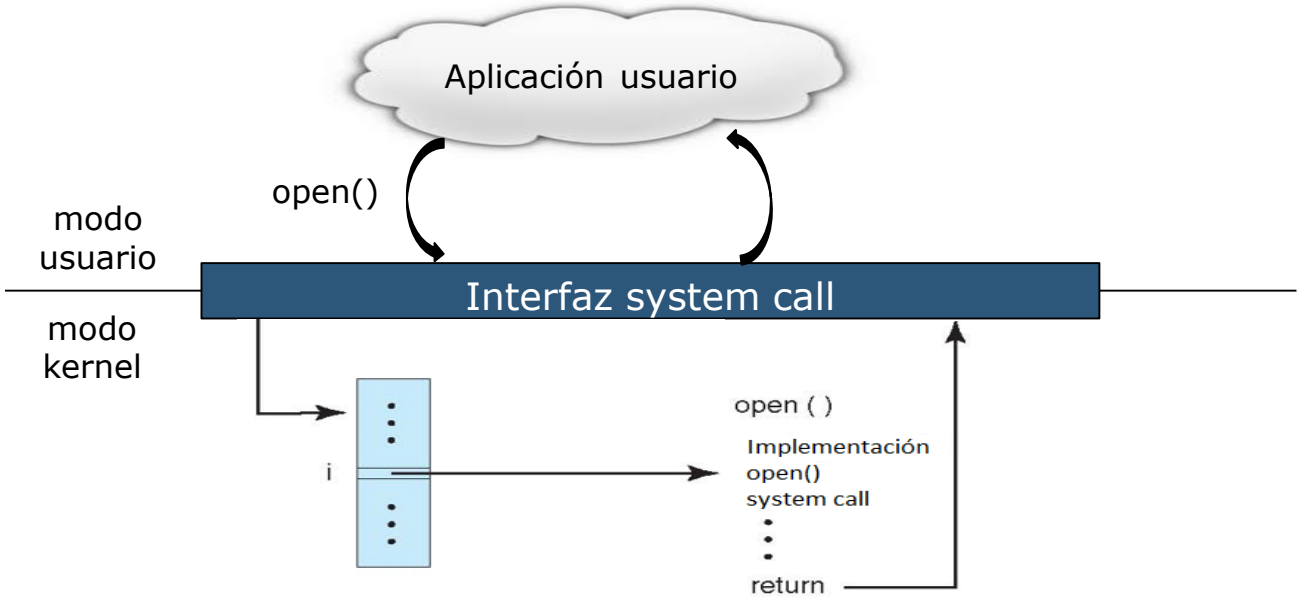
iOS

AGENDA

1. Conceptos
   1. Servicios de Sistemas operativos
   2. Interfaz de Usuario del Sistema Operativo
2. **Llamadas a Sistema**
   1. **Tipos de Llamadas a Sistema**
   2. **Pasaje de Parámetros**
3. Programas de Sistemas
4. Diseño, Implementación y Estructura de un Sistema Operativo
5. Generación y Boot del Sistema
6. Conceptos de Máquinas Virtuales

LLAMADAS AL SISTEMA

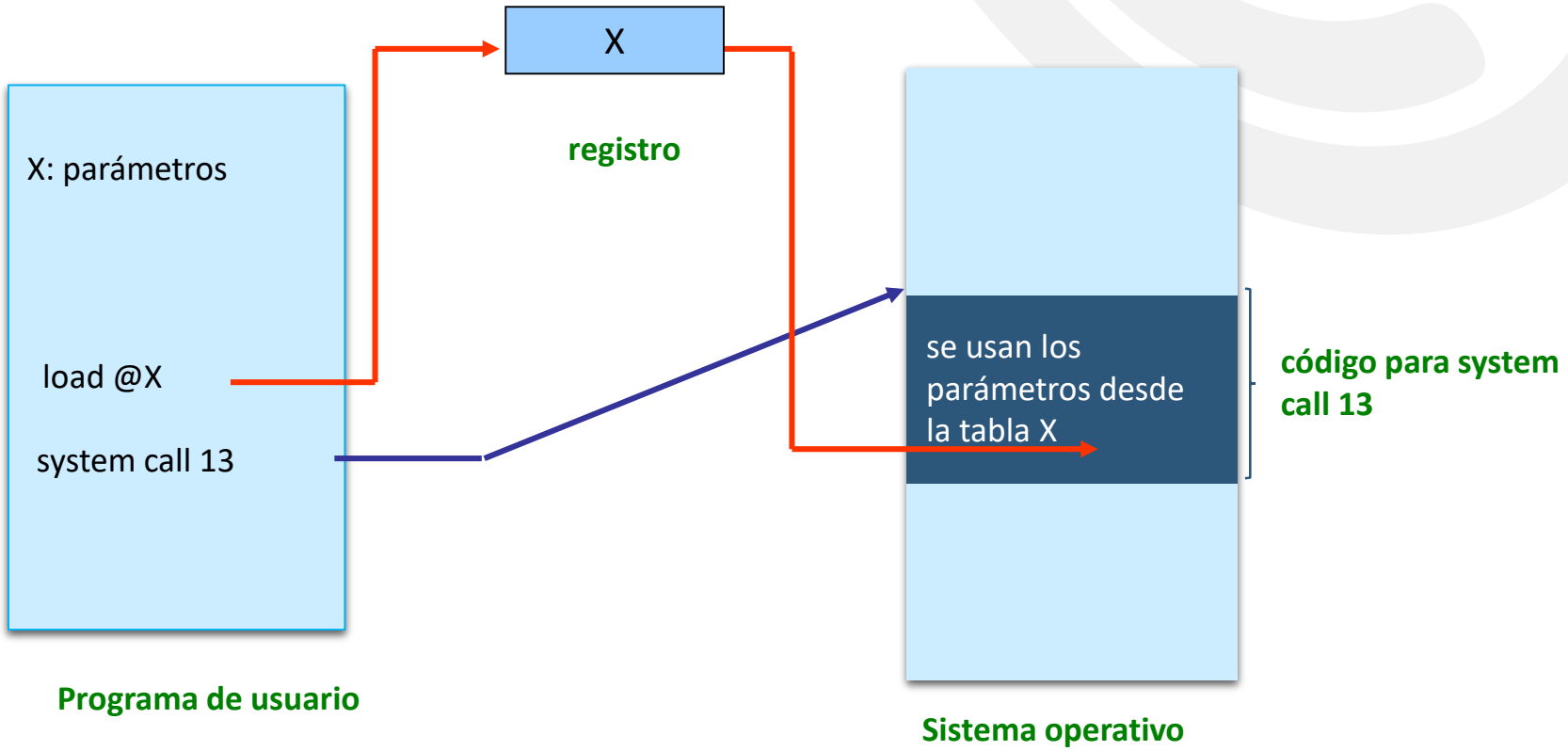
- Son la interfaz de programación a los servicios provistos por el SO
- Típicamente escritas en lenguajes de alto nivel (C o C++)
- Mayoritariamente accedidas por programas vía *Application Program Interface (API)* más que por el uso llamadas a sistema directas



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.

PASAJE DE PARÁMETROS PASADOS VÍA TABLA



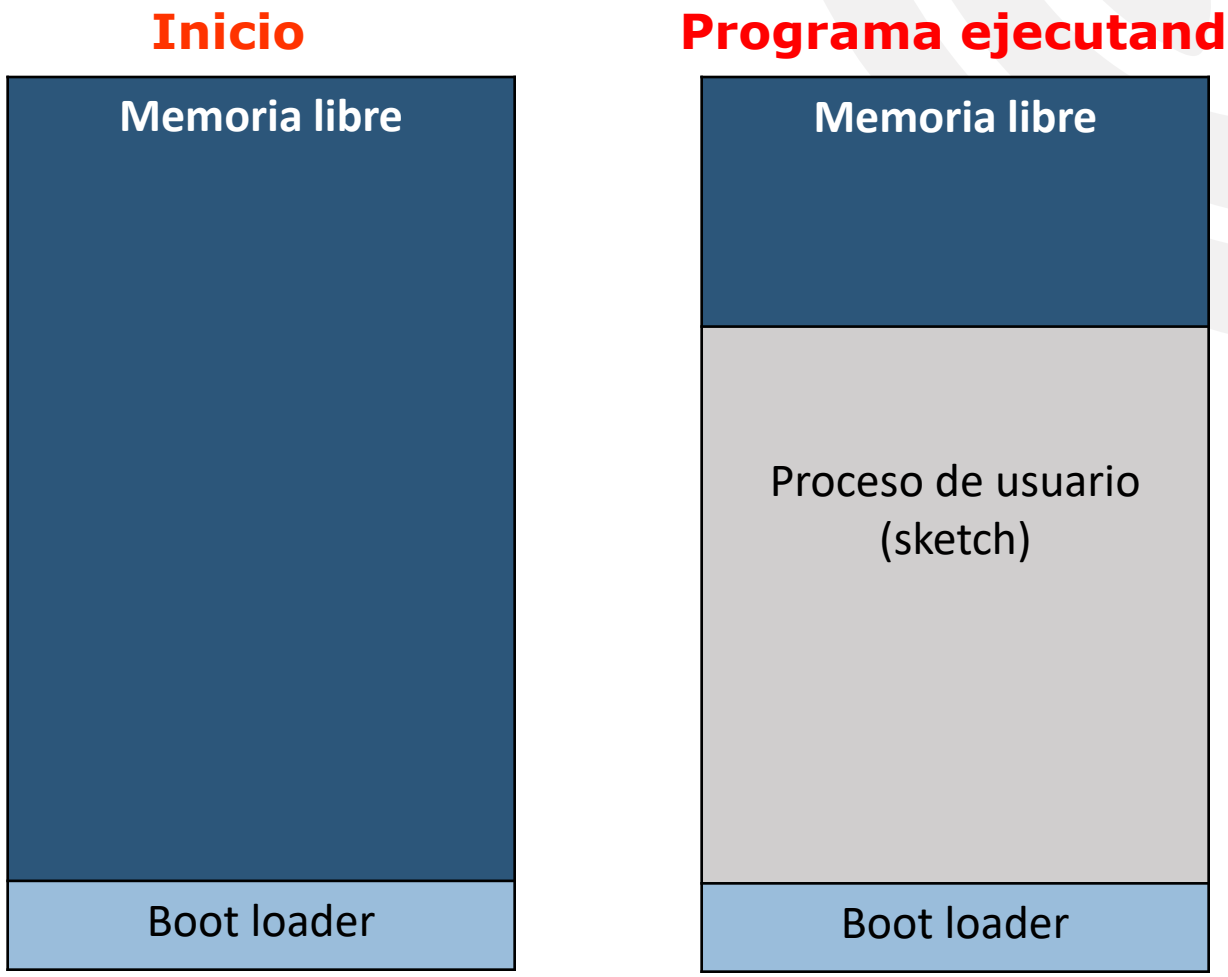
TIPOS LLAMADAS A SISTEMA

- 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

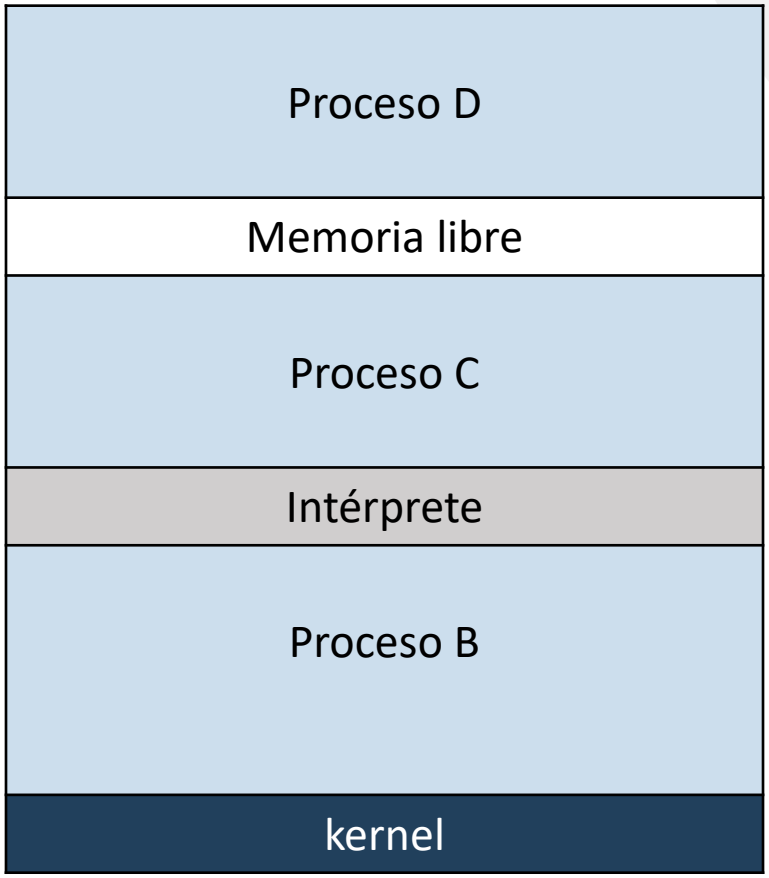
EJEMPLOS DE LLAMADAS A SISTEMA

	Windows	UNIX
Procesos	CreateProcess() ExitProcess() WaitForSingleObject()	Fork() Exit() Wait()
Archivos	CreateFile() ReadFile() WriteFile() CloseHandle()	Open() Read() Write() Close()
Dispositivos	SetConsoleMode() ReadConsole() WriteConsole()	Ioctl() Read() Write()
Información	GetCurrentProcessID() SetTimer() Sleep()	Getpid() Alarm() Sleep()
Comunicación	CreatePipe() CreateFileMapping() MapViewOfFile()	Pipe() Shmget() Mmap()
Protección	SetFileSecurity() InitIializeSecurityDescriptor() SetSecutrityDescriptorGroup()	Chmod() Umask() Chown()

UNA TAREA: EJEMPLO DE EJECUCIÓN EN ARDUINO



MÚLTIPLES TAREAS: EJEMPLO EJECUCIÓN EN FREEBSD



AGENDA

1. Conceptos
    1. Servicios de Sistemas operativos
    2. Interfaz de Usuario del Sistema Operativo
2. Llamadas a Sistema
    1. Tipos de Llamadas a Sistema
    2. Pasaje de Parámetros
3. **Programas de Sistemas**
4. Diseño, Implementación y Estructura de un Sistema Operativo
5. Generación y Boot del Sistema
6. Conceptos de Máquinas Virtuales

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).

APLICACIONES Y SISTEMAS OPERATIVOS

- Las aplicaciones compiladas en un sistema generalmente no se pueden ejecutar en otros sistemas operativos
- Cada Sistema operativo tiene sus propias llamadas al sistema
  - Formatos de archivo propios, etc.
- Las aplicaciones pueden ser para multiples sistemas operativos
  - Escritas en un lenguaje interpretado como like Python, Ruby, y con intérprete disponible en múltiples sistemas operativos
  - Aplicación escrita en un lenguaje que incluye una máquina virtual que contiene la aplicación en ejecución (como Java)
  - Utilizar un lenguaje standard (como C), compile separadamente en cada sistemas operative para poder ejecutarla

AGENDA

1. Conceptos
    1. Servicios de Sistemas operativos
    2. Interfaz de Usuario del Sistema Operativo
2. Llamadas a Sistema
    1. Tipos de Llamadas a Sistema
    2. Pasaje de Parámetros
3. Programas de Sistemas
4. **Diseño, Implementación y Estructura de un Sistema Operativo**
5. Generación y Boot del Sistema
6. Conceptos de Máquinas Virtuales

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

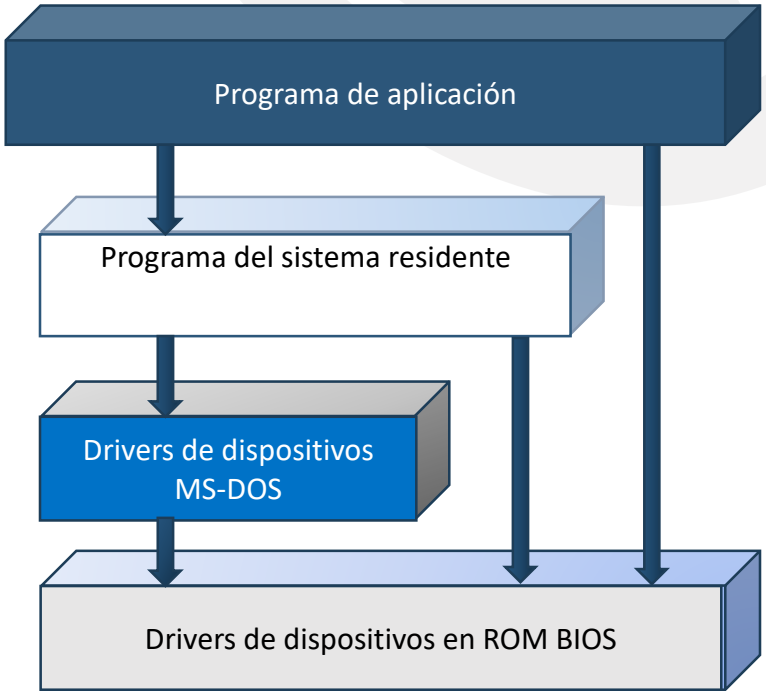
  Política: ¿Qué deberá hacerse?
  Mecanismo: ¿Cómo hacerlo?

- 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

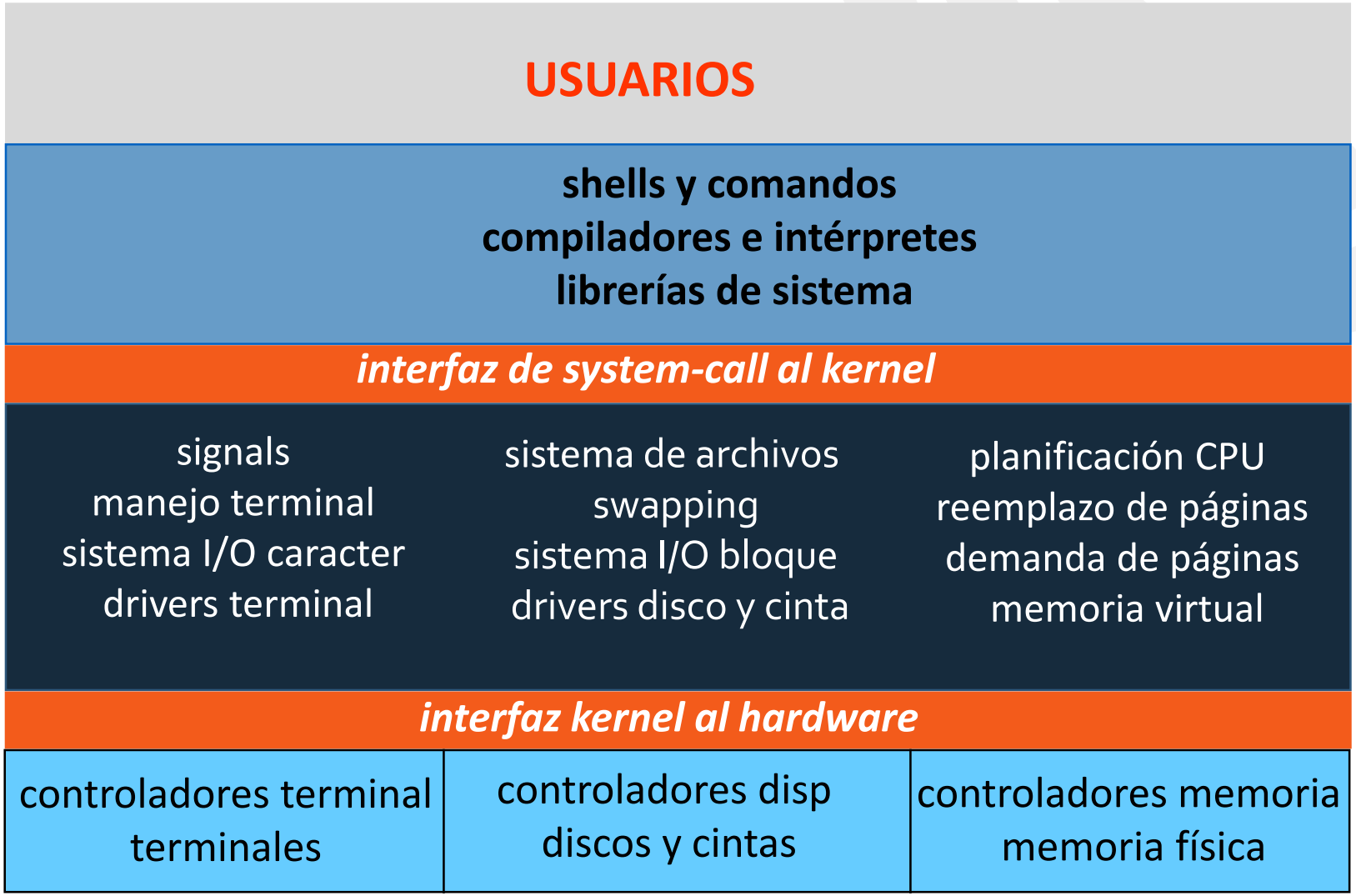
Caso UNIX

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

El SO UNIX consiste de dos partes separables.

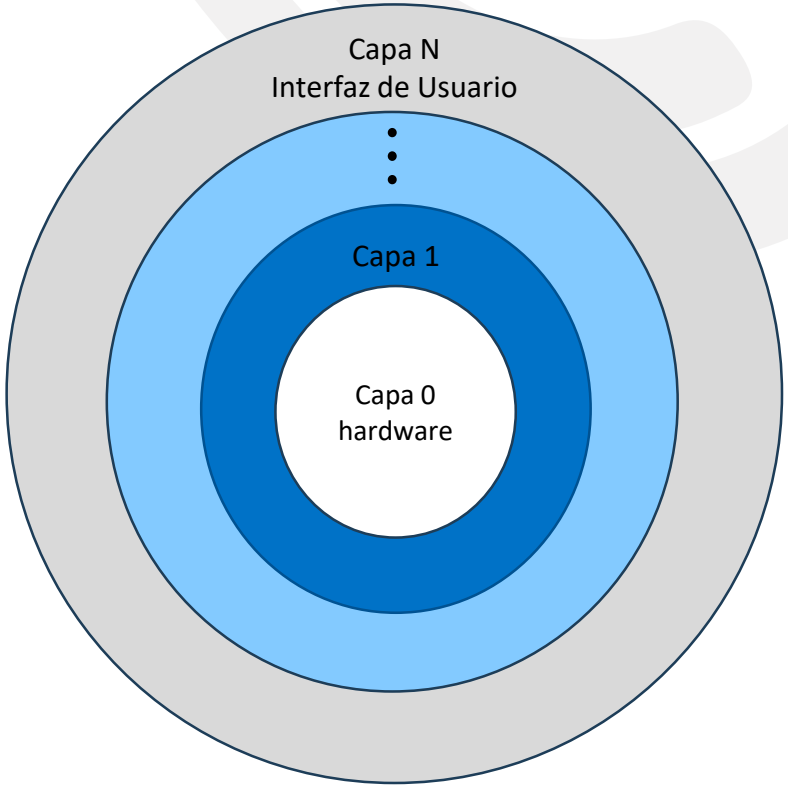
- Programas de sistema
- El kernel
  - Consiste de todo lo que esta debajo de la interfaz de los system calls y encima del hardware
  - Contiene el sistema de archivos, la planificación de CPU, manejo de memoria, y otras funciones del sistema operativo; un gran número de funciones en un solo nivel.

ESTRUCTURA SIMPLE - UNIX



ENFOQUE POR CAPAS

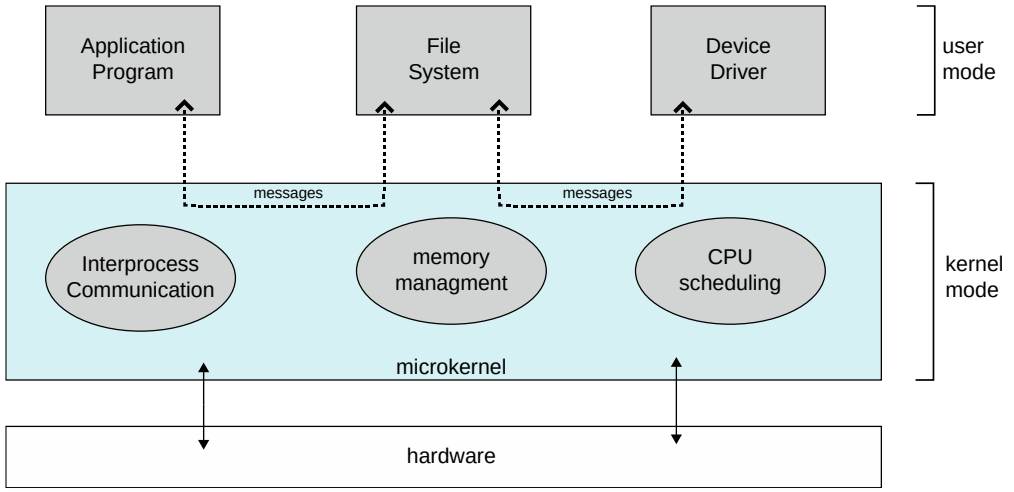
- El sistema operativo está dividido en un número de capas (niveles), cada una construída sobre el tope de otra. La capa inferior (nivel 0), es el hardware; la más alta (capa N) es la interfaz de usuario.
- En forma modular, las capas son seleccionadas de manera que cada una usa funciones (operaciones)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



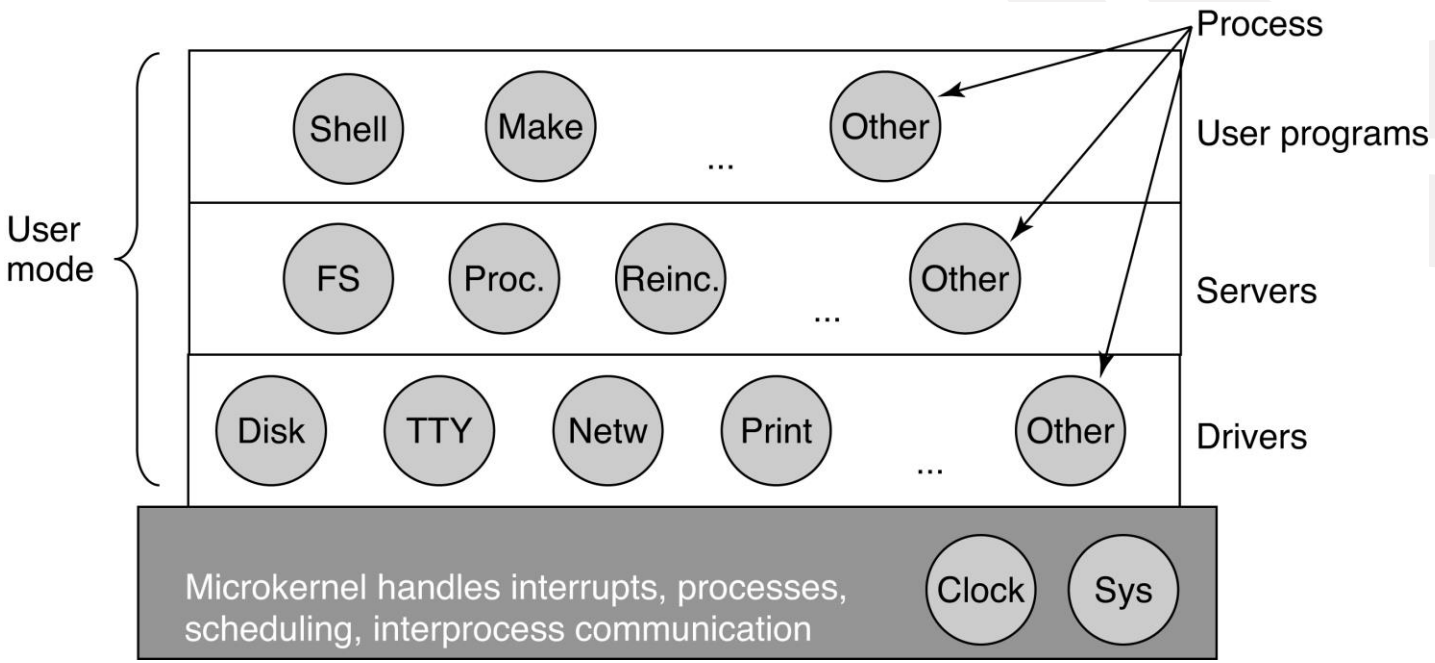
ESTRUCTURA DE SISTEMA MICROKERNEL

- 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

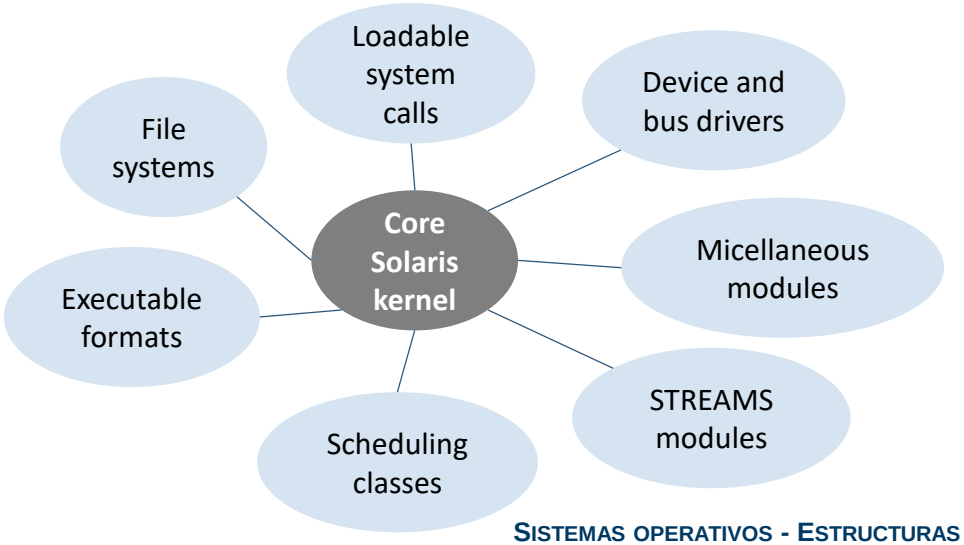


SISTEMA MICROKERNEL – EJEMPLO: MINIX 3



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

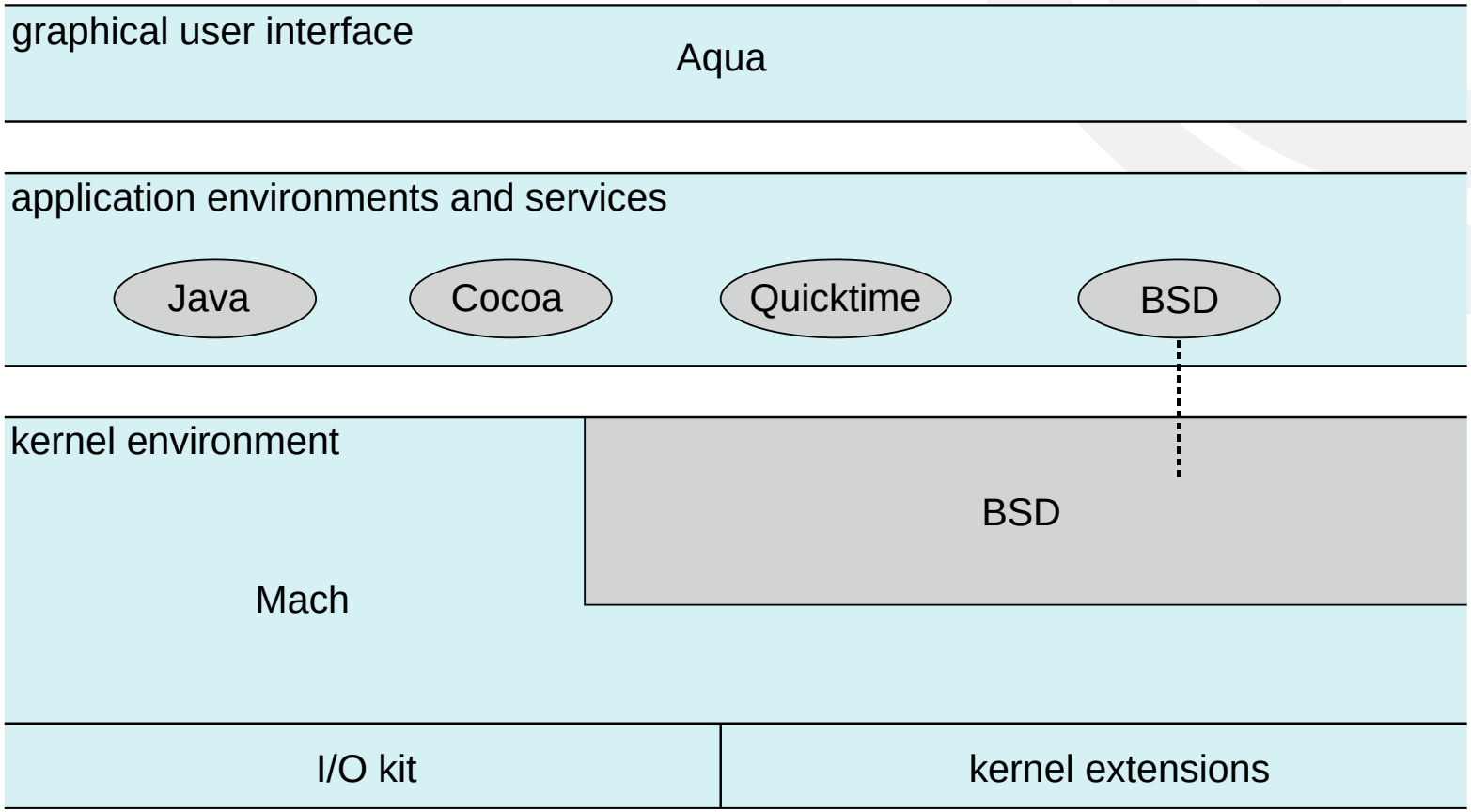


SISTEMAS HÍBRIDOS

- Los sistemas operativos modernos no presentan un modelo puro.

- Los modelos híbridos combinan multiples aproximaciones para alcanzar rendimento, 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)

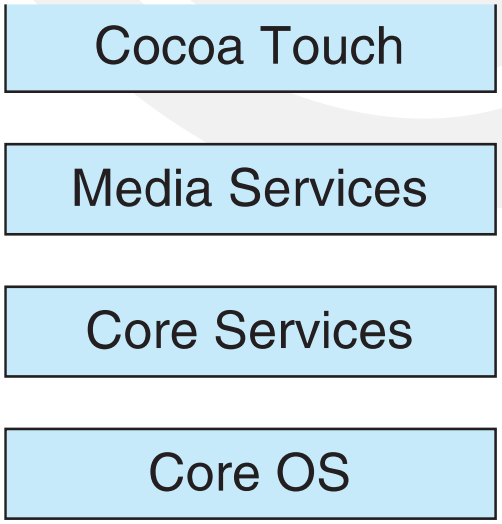
SISTEMAS HÍBRIDOS - ESTRUCTURA DE MAC OS X



SISTEMAS HÍBRIDOS - IOS

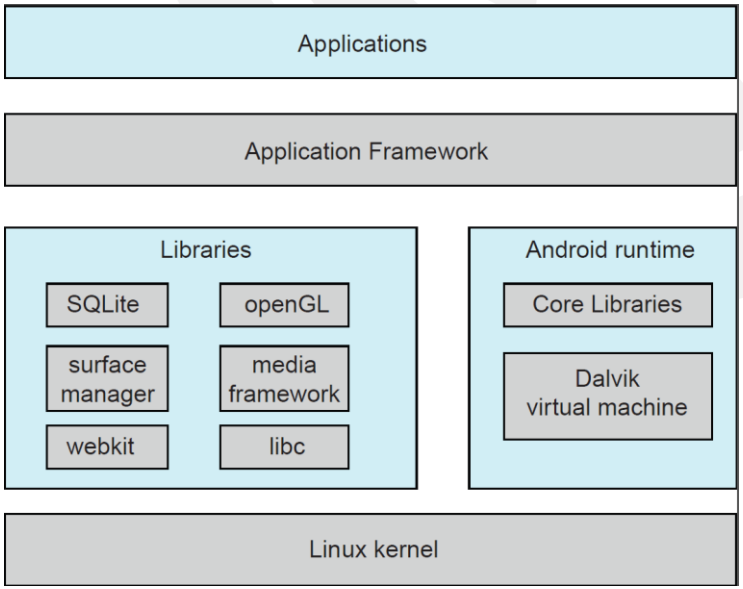
SO de Apple móvil para iPhone, iPad

- 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

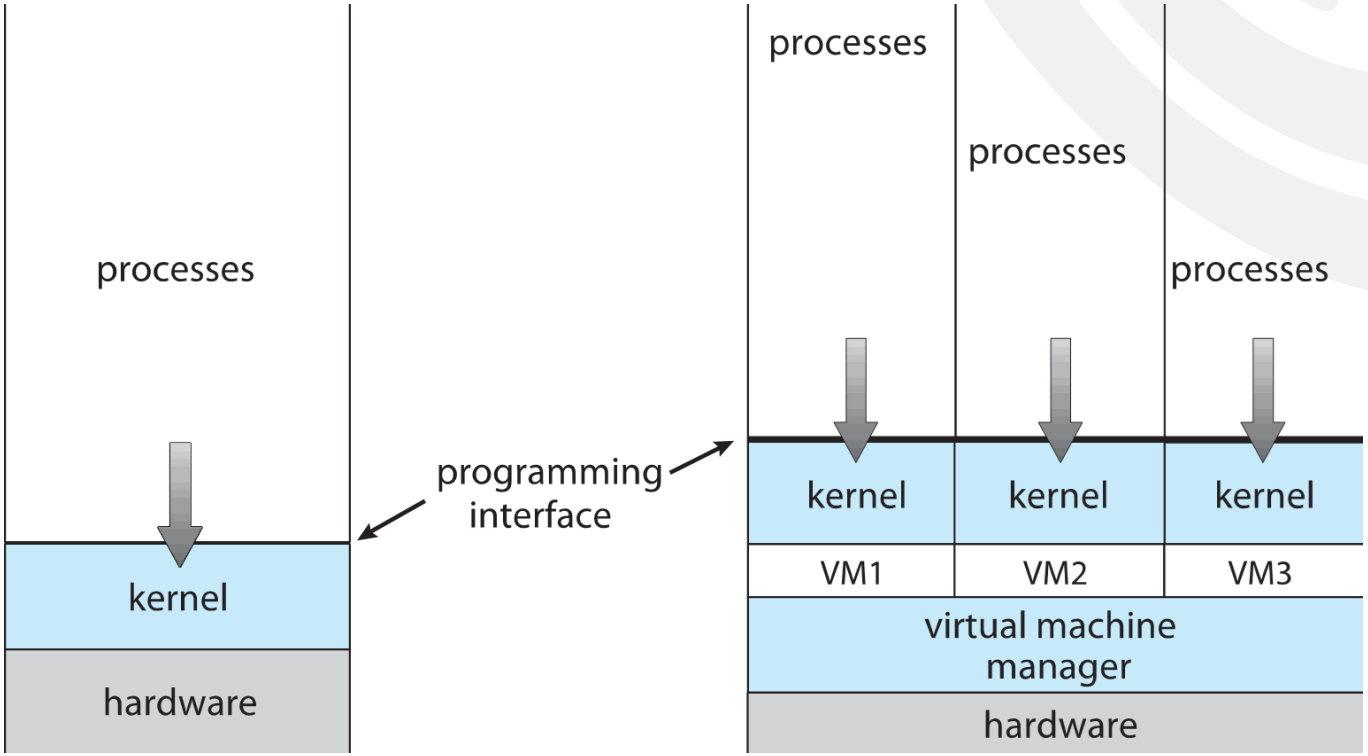
¿POR QUÉ VIRTUALIZAR?

- Reduce el costo e incrementa la eficiencia de los existentes recursos de hardware
- Lograr más en menos tiempo

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.

MÁQUINAS VIRTUALES



Máquina no virtual

Máquina virtual

MÁQUINAS VIRTUALES – VMM (VIRTUAL MACHINE MANAGER)

Virtual Machine Manager

Crea, administra y ejecuta las máquinas virtuales.

Clasificación

- Tipo 0 – son soluciones basados en hardware, que proveen soporte para la creación y administración via el firmware.
- Tipo 1 – Hypervisors ejecutan directamente sobre el hardware de la máquina.
- Tipo 2 – Hypervisors ejecutan sobre el sistema operativo host que provee los servicios de virtualización.

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.