Estructuras
Sistemas Operativos

Estructura Sistemas Operativos

- Servicios del Sistema Operativo
- Interfaz del Sistema Operativo del usuario
- Llamadas del Sistema
- Tipos de llamadas al Sistema
- Programas del Sistema
- Diseño e implementación del Sistema Operativo
- Estructura del Sistema Operativo
- Depuración del Sistema Operativo
- Inicio del Sistema

Objectivo

- Describir los servicios que un sistema operativo proporciona a usuarios, procesos y a otros sistemas.
- Discutir las diversas formas de estructurar un sistema operativo.
- Explicar cómo se instalan y personalizan los sistemas operativos y cómo se inician.

Servicios Sistema Operativo

Los sistemas operativos proporcionan un entorno para la ejecución de programas, servicios a programas y usuarios.

Un conjunto de servicios del sistema operativo proporciona funciones que son útiles para el usuario:

- **Interfaz de usuario** - Casi todos los sistemas operativos tienen una interfaz de usuario (UI).
  - Varía entre **Command-Line (CLI)**, **Graphics User Interface (GUI)**, **Batch**
- **Ejecución del programa** - El sistema debe ser capaz de cargar un programa en la memoria y ejecutar ese programa, finalizar la ejecución, normalmente o anormalmente (indicando error).
- **Operaciones de I/O** - Un programa en ejecución puede requerir I/O, que puede incluir un archivo o un dispositivo de I/O.

Servicios Sistema Operativo

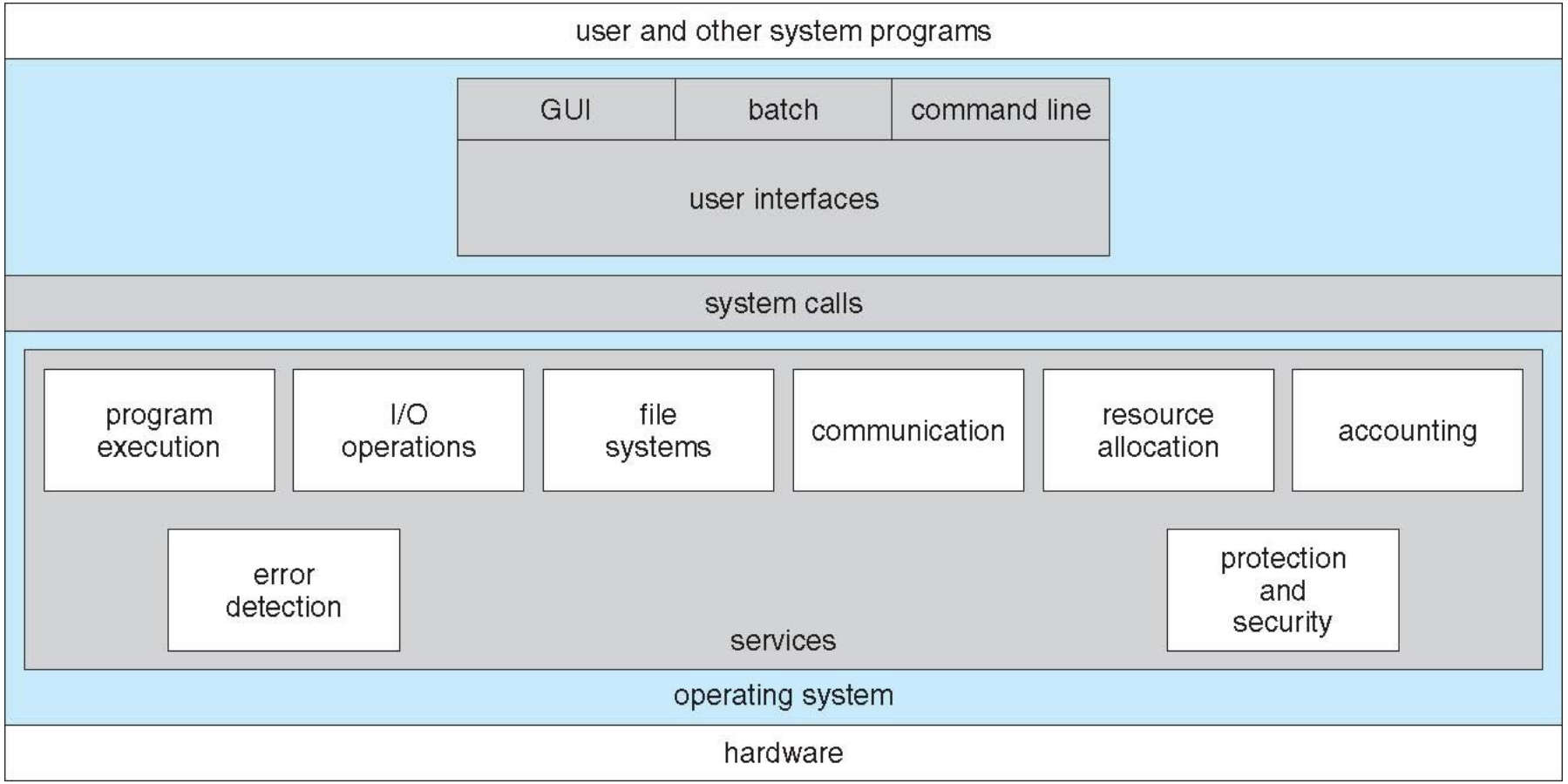
- **Manipulación del sistema de archivos** - El sistema de archivos es de particular interés. Los programas necesitan leer y escribir archivos y para los directorios: crearlos, borrarlos, buscarlos, listar información de archivos y la administración de permisos.
- **Comunicaciones** - Los procesos pueden intercambiar información, en la misma computadora o entre computadoras a través de una red
  - Las comunicaciones pueden ser a través de memoria compartida o mediante el paso de mensajes (paquetes movidos por el sistema operativo)
- **Detección de errores** - El sistema operativo debe estar constantemente consciente de posibles errores
  - Puede ocurrir en la CPU y el hardware de memoria, en dispositivos de I/O, en el programa de usuario
  - Para cada tipo de error, el sistema operativo debe tomar las medidas adecuadas para garantizar una computación correcta y coherente
  - El uso de la depuración pueden mejorar en gran medida las capacidades del usuario y del programador para utilizar eficientemente el sistema.

Servicios Sistema Operativo

Existe otro conjunto de funciones del SO para asegurar el funcionamiento eficiente del mismo sistema a través del uso compartido de recursos

- **Asignación de recursos**: cuando varios usuarios o varios trabajos se ejecutan simultáneamente, los recursos deben asignarse a cada uno de ellos
  - Muchos tipos de recursos: ciclos de CPU, memoria principal, almacenamiento de archivos, dispositivos de I/O.
- **Contabilidad** - Para realizar un seguimiento de qué usuarios utilizan cuánto y qué tipo de recursos de la computadora
- **Protección y seguridad** - Los propietarios de información almacenada en un sistema informático multiusuario o en red pueden querer controlar el uso de esa información, los procesos concurrentes no deben interferir entre sí
  - **Protección** implica asegurar que todo el acceso a los recursos del sistema está controlado
  - **Seguridad** del sistema requiere autenticación de usuario, se extiende a dispositivos de I/O externas para proteger de intentos de acceso no válidos.

Los servicios del sistema operativo

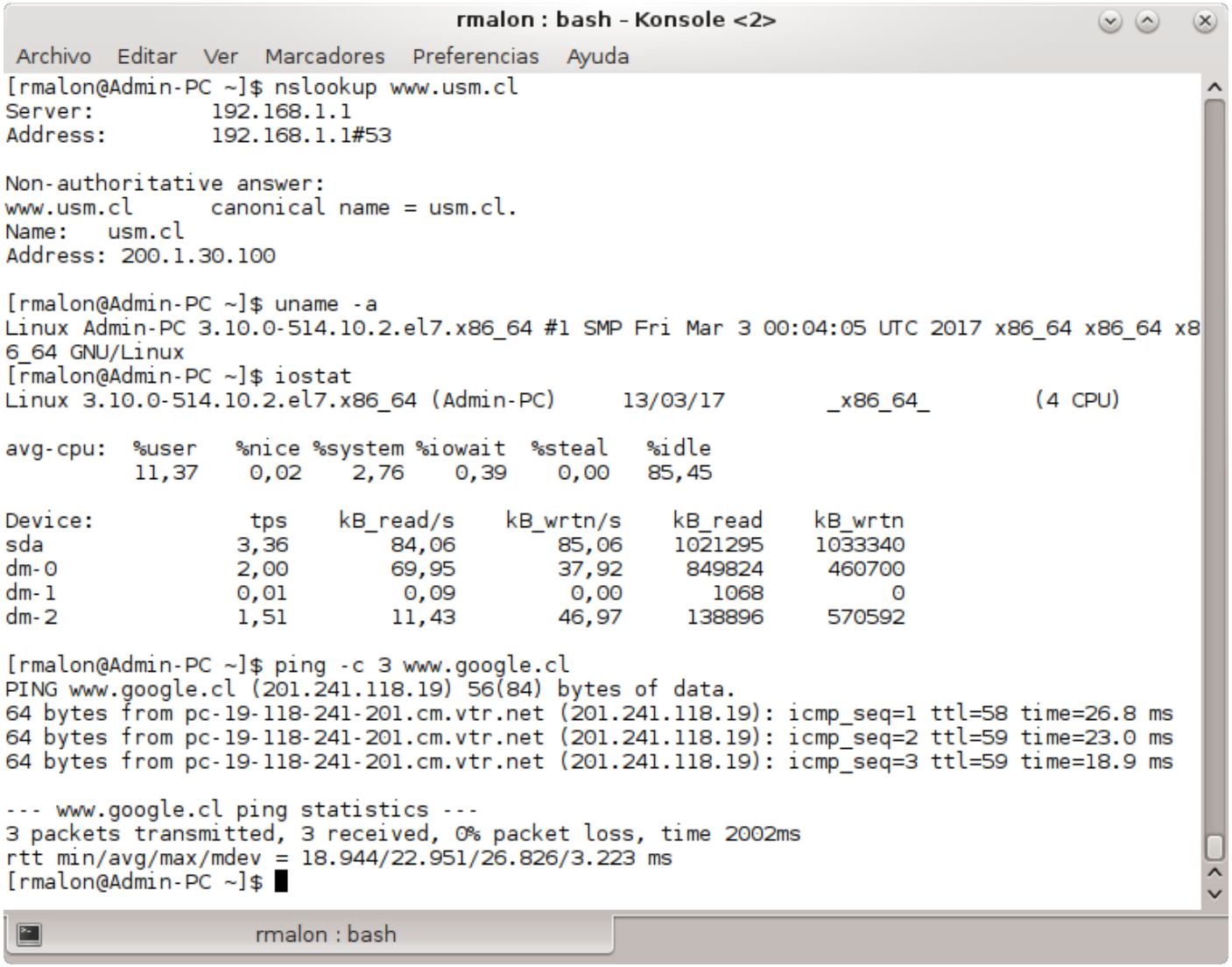


Interfaz del sistema operativo del usuario - CLI

CLI O el intérprete de comandos permite la entrada directa de comandos.

- A veces implementado en kernel, a veces por programa de sistemas.
- Se implementan múltiples interpretes que se conocen como – shells
- Función Principal, primero busca un comando del usuario y lo ejecuta
- Se implementa la mayoría de los comandos a tráves de una serie de programas de sistema

Bourne Shell Interprete de Comando



Interfaz del sistema operativo del usuario - GUI

- Interfaz de escritorio de uso fácil
  - Normalmente ratón, teclado y monitor
  - Los iconos representan archivos, programas, acciones, etc.
  - Varios botones del ratón sobre los objetos de la interfaz causan varias acciones (proporcionar información, opciones, ejecutar la función, abrir el directorio (conocido como una carpeta)
  - Inventado en Xerox PARC
- Muchos sistemas ahora incluyen interfaces CLI y GUI
  - Microsoft Windows es GUI con CLI "cmd" shell
  - Apple Mac OS X es "Aqua" interfaz gráfica de usuario con el kernel UNIX debajo y shells disponibles
  - Unix y Linux tienen CLI con interfaces GUI opcionales (CDE, KDE, GNOME)

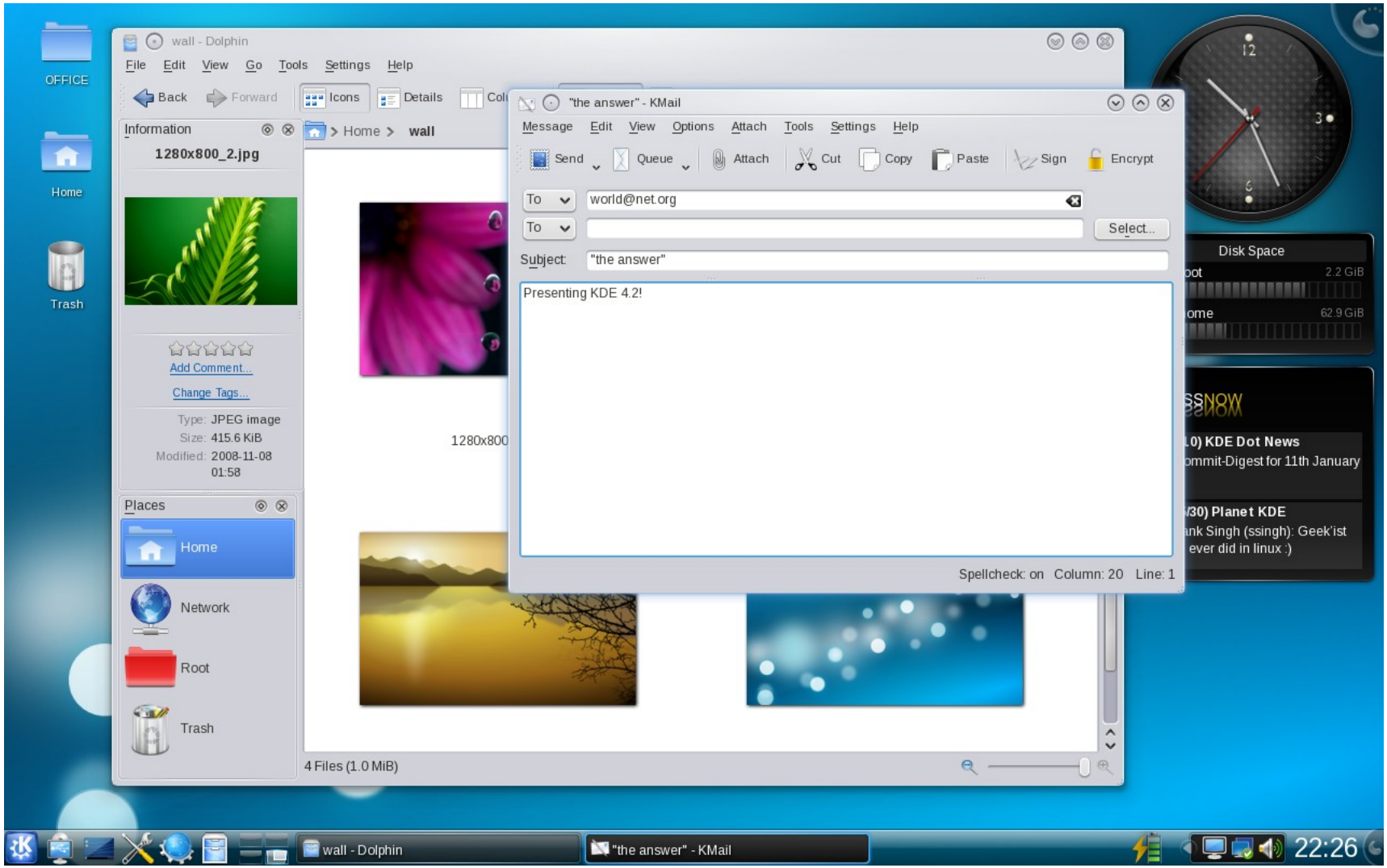
Interfaces de pantalla táctil

Los dispositivos de pantalla táctil requieren nuevas interfaces

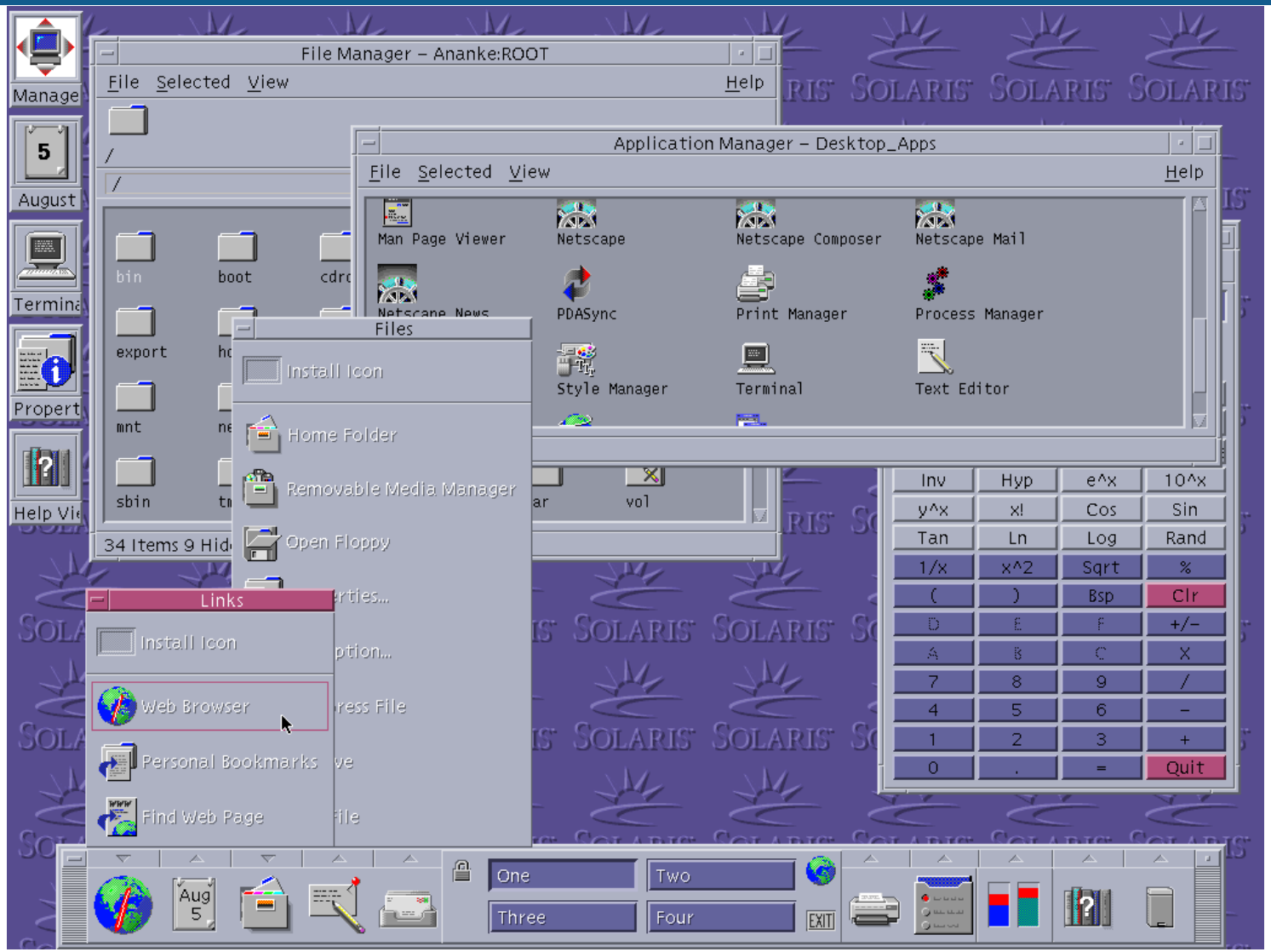
- Ratón no posible o no deseado
- Acciones y selección basada en gestos
- Teclado virtual para entrada de texto
- Comandos de voz.

The Mac OS X GUI

KDE GUI


wall - Dolphin
File Edit View Go Tools Settings Help
Back Forward Icons Details
Information
1280x800_2.jpg
Add Comment...
Change Tags...
Type: JPEG image
Size: 415.6 KiB
Modified: 2008-11-08 01:58
Places
Home
Network
Root
Trash
Home > wall
4 Files (1.0 MiB)
"the answer" - KMail
Message Edit View Options Attach Tools Settings Help
Send Queue Attach Cut Copy Paste Sign Encrypt
To world@net.org
To
Select...
Subject: "the answer"
Presenting KDE 4.2!
Spellcheck: on Column: 20 Line: 1
Disk Space
2.2 GiB
62.9 GiB
KDE Dot News
Planet KDE
wall - Dolphin
"the answer" - KMail
22:26

CDE GUI


File Manager – Ananke:ROOT
File Selected View Help
Application Manager – Desktop_Apps
File Selected View Help
Man Page Viewer
Netscape
Netscape Composer
Netscape Mail
PDASync
Print Manager
Process Manager
Style Manager
Terminal
Text Editor
Files
Install Icon
Home Folder
Removable Media Manager
Open Floppy
Links
Install Icon
Web Browser
Personal Bookmarks
Find Web Page
Inv Hyp e^x 10^x
y^x x! Cos Sin
Tan Ln Log Rand
1/x x^2 Sqrt %
( ) Bsp Clr
A B C D E F +/-
7 8 9 /
4 5 6 -
1 2 3 +
0 . = Quit
One Two
Three Four
EXIT
Aug 5

GNOME GUI

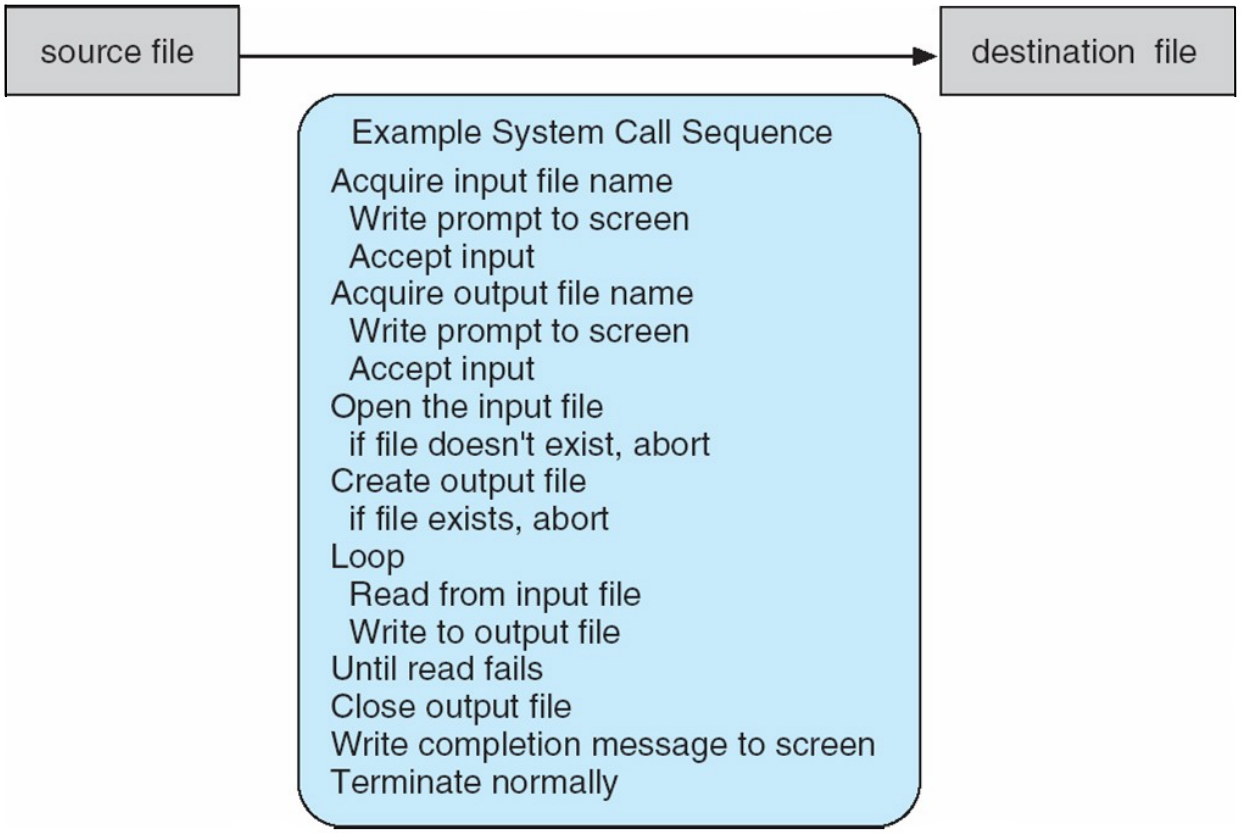

Activities
Wed 13:49
Type to search...
Boxes
Builder
Calendar
Cheese
Clocks
Contacts
Devhelp
Documents
Files
Firefox
gitg
GNU Image Man...
Inkscape
LibreOffice Calc
LibreOffice Draw
LibreOffice Impr...
LibreOffice Writer
Maps
Nibbles
Notes
Photos
Polari
Rhythmbox
Settings
Frequent
All

Llamadas a Sistema

- Interfaz de programación para los servicios proporcionados por el sistema operativo.
- Normalmente escrito en un lenguaje de alto nivel (C o C ++).
- Se accede principalmente a los programas a través de una Interfaz de Programación de Aplicación (API) de alto nivel en lugar del uso directo de la llamada del sistema
- Las tres API más comunes son API Win32 para Windows, API POSIX para sistemas basados en POSIX (incluyendo prácticamente todas las versiones de UNIX, Linux y Mac OS X) y API Java para la máquina virtual Java (JVM)

Ejemplo de Llamada a Sistema

Secuencia de llamada del sistema para copiar el contenido de un archivo a otro archivo



Ejemplo de API estandar

EXAMPLE OF STANDARD API

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

```
#include <unistd.h>

ssize_t     read(int fd, void *buf, size_t count)
```

return value — function name — parameters

A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

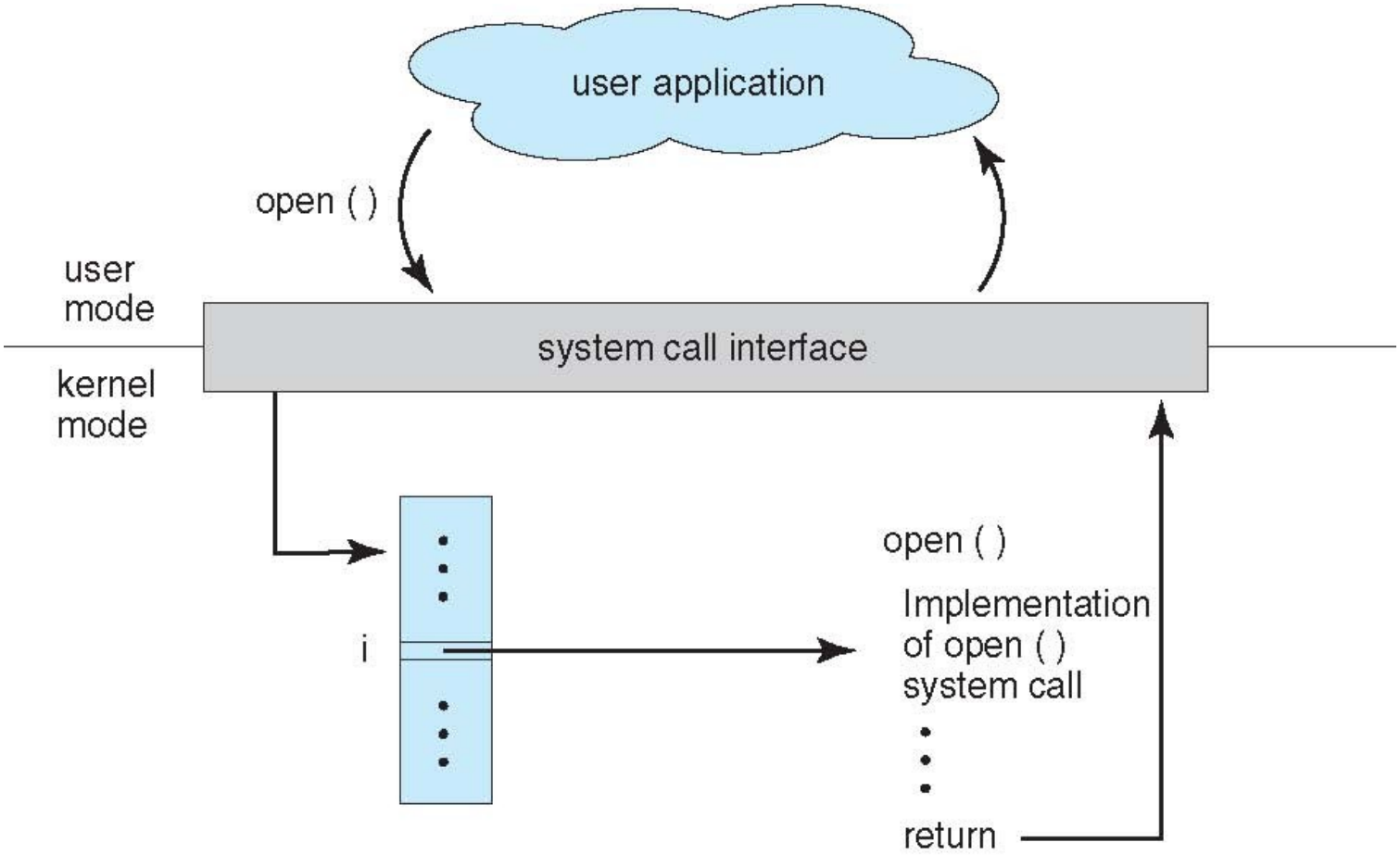
- `int fd`—the file descriptor to be read
- `void *buf`—a buffer where the data will be read into
- `size_t count`—the maximum number of bytes to be read into the buffer

On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, `read()` returns −1.

Implementación Llamada Sistema

- Número asociado con cada llamada de sistema
  - **System-call interface** Mantiene una tabla indexada de acuerdo a estos números
- La interfaz de llamada del sistema invoca la llamada de sistema prevista en el kernel del sistema operativo y devuelve el estado de la llamada del sistema y cualquier valor de retorno
- El que llama no necesita saber nada acerca de cómo se implementa la llamada al sistema
  - Sólo tiene que obedecer API y entender lo que el sistema operativo hará como resultado llamada
  - La mayoría de los detalles de la interfaz de OS ocultos al programador por API
    - Gestionado por la biblioteca de soporte técnico en tiempo de ejecución (conjunto de funciones integradas en las bibliotecas incluidas con el compilador)

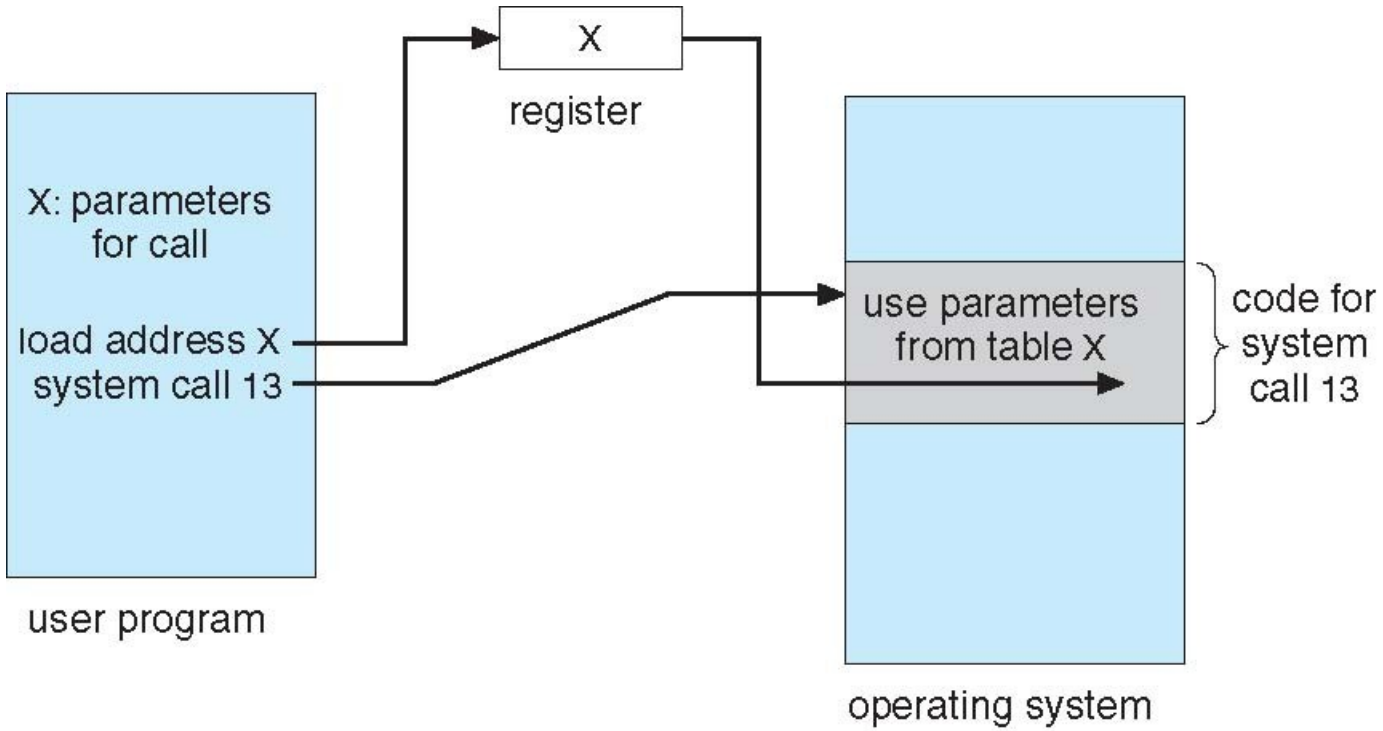
API – Llamada al sistema - relación OS



Paso del parámetro de llamada del sistema

- A menudo, se requiere más información que simplemente la identidad de la llamada de sistema deseada
  - El tipo exacto y la cantidad de información varían según el sistema operativo y la llamada
- Tres métodos generales utilizados para pasar parámetros al sistema operativo
  - Simplest: pasa los parámetros en los registros
    - En algunos casos, pueden ser más parámetros que los registros
  - Parámetros almacenados en un bloque, o tabla, en memoria, y dirección del bloque pasado como un parámetro en un registro
    - Este enfoque adoptado por Linux y Solaris
  - Parámetros colocados o empujados en la pila por el programa y sacados de la pila por el sistema operativo
  - Los métodos de bloque y pila no limitan el número o la longitud de los parámetros que se pasan.

Parámetro que pasa a través de la tabla



Tipos de Llamadas al Sistema

- Control de procesos
  - Crear proceso, finalizar proceso
  - Terminar, abortar
  - Cargar, ejecutar
  - Obtener atributos de proceso, establecer atributos de proceso
  - Esperar el tiempo
  - Evento de espera, evento de señal
  - Asignar y liberar memoria
  - Memoria de volcado si hay error
  - Depurador para determinar errores, ejecución de un solo paso
  - Bloqueos para gestionar el acceso a datos compartidos entre procesos

Tipos de Llamadas al Sistema

- Gestión de archivos
  - Crear archivo, borrar archivo
  - Abrir, cerrar archivo
  - Leer, escribir, reposicionar
  - Obtener y establecer atributos de archivo
- Gestión de dispositivos
  - Dispositivo de solicitud, dispositivo de liberación
  - Leer, escribir, reposicionar
  - Obtener atributos de dispositivo, establecer atributos de dispositivo
  - Conectar o desacoplar lógicamente dispositivos

Tipos de Llamadas al Sistema

- Mantenimiento de la información
  - Obtener la hora o fecha, establecer la hora o la fecha
  - Obtener datos del sistema, configurar los datos del sistema
  - Obtener y configurar los atributos de proceso, archivo o dispositivo
- Comunicaciones
  - Crear, eliminar conexión de comunicación
  - Enviar, recibir mensajes si el modelo de paso de mensajes a nombre de host o nombre de proceso
    - De cliente a servidor
  - El modelo de memoria compartida crea y obtiene acceso a regiones de memoria
  - Información sobre el estado de la transferencia
  - Adjuntar y desconectar dispositivos remotos

Tipos de Llamadas al Sistema

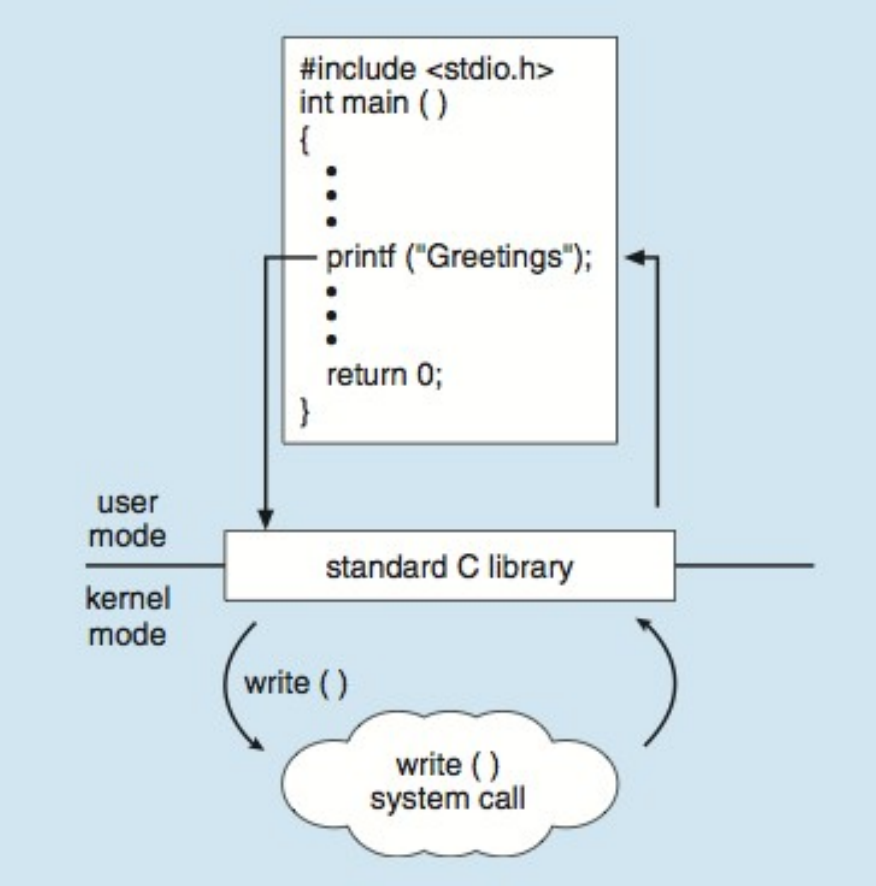
- Proteccion
  - Controlar el acceso a los recursos
  - Obtener y establecer permisos
  - Permitir y denegar el acceso de usuario

Ejemplo de Llamadas a Sistema Windows y Unix

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

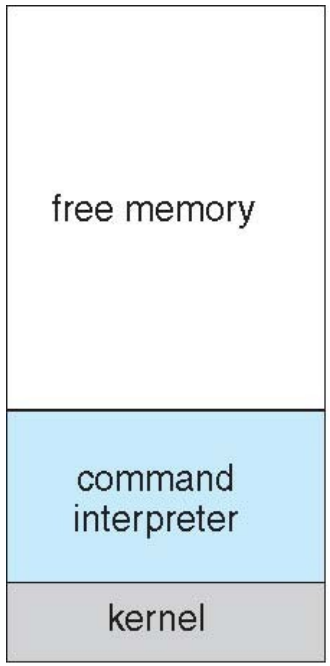
Ejemplo Librería estandar C

Programa C que invoca printf () llamada de biblioteca, que llama write () que es una llamada de sistema



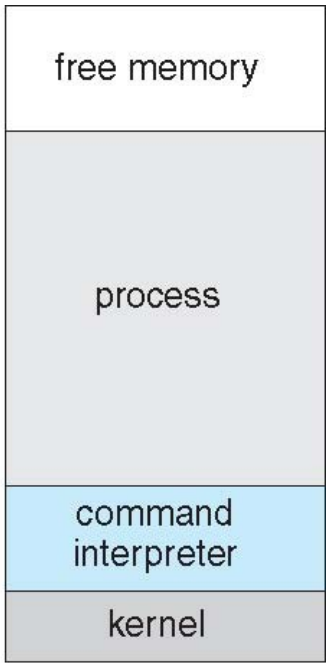
Ejemplo: MS-DOS

- Tarea única
- Shell se invoca cuando se inicia el sistema
- Método simple para ejecutar el programa
  - Ningún proceso creado
- Espacio de memoria único
- Carga el programa en la memoria, sobrescribiendo todo, excepto el kernel
- Salida del programa -> shell recargado



(a)

At system startup



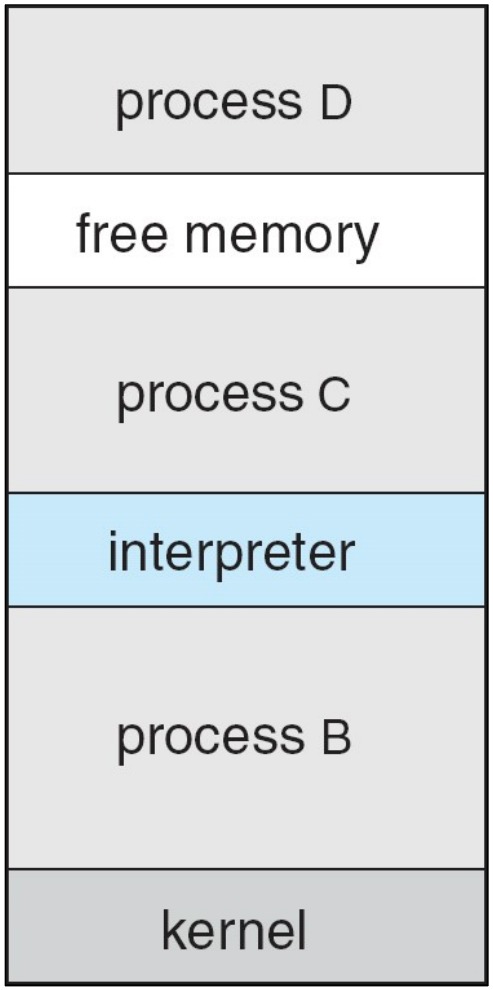
(b)

running a program

Ejemplo: FreeBSD

- Variante Unix
- Multitarea
- Inicio de sesión de usuario -> invocar la elección del usuario de shell
- Shell ejecuta fork () llamada del sistema para crear proceso

- El proceso sale con:
  - Code = 0 - sin error
  - Código> 0 - código de error



Programas del Sistema

- Los programas del sistema proporcionan un entorno conveniente para el desarrollo y la ejecución del programa. Pueden dividirse en:
  - Manipulación de archivos
  - Información de estado a veces almacenada en una modificación de archivo
  - Soporte de lenguaje de programación
  - Carga y ejecución del programa
  - Comunicaciones
  - Servicios de Background
  - Programas de aplicación

La visión de la mayoría de los usuarios del sistema operativo se define por los programas del sistema, no las llamadas reales del sistema.

Programas de Sistema

- Proporcionar un ambiente conveniente para el desarrollo y ejecución de programas
  - Algunos de ellos son simplemente interfaces de usuario para llamadas al sistema; Otros son considerablemente más complejos
- Administración de archivos - Crear, eliminar, copiar, renombrar, imprimir, lista y en general manipular archivos y directorios
- Información de Estado
  - Información de sistema - fecha, hora, cantidad de memoria disponible, espacio en disco, número de usuarios
  - Información detallada de rendimiento, registro y depuración
  - Normalmente, estos programas formatean e imprimen la salida al terminal u otros dispositivos de salida
  - Algunos sistemas implementan un registro - utilizado para almacenar y recuperar información de configuración

Programas de Sistema

- **Modificación del archivo**
  - Editores de texto para crear y modificar archivos
  - Comandos especiales para buscar contenido de archivos o realizar transformaciones del texto
- **Soporte de lenguaje de programación** - Compiladores, ensambladores, depuradores e intérpretes (C, C++, Java, PERL).
- **Carga y ejecución de programas** - Cargadores absolutos, cargadores reubicables, editores de montaje y cargadores de sustitución, sistemas de depuración para lenguaje de alto nivel y máquina.
- **Comunicaciones** - Proporcionar el mecanismo para crear conexiones virtuales entre procesos, usuarios y sistemas informáticos
  - Permite a los usuarios enviar mensajes a las pantallas de los demás, navegar por páginas web, enviar mensajes de correo electrónico, iniciar sesión remotamente, transferir archivos de una máquina a otra

Programas de Sistema

- Servicios de background
  - Lanzamiento en el arranque
    - Algunos para el inicio del sistema, luego finalizan.
    - Algunos desde el arranque del sistema hasta el apagado.
  - Proporcionan comprobación de disco, programación de procesos, registro de errores, impresión
  - Se Ejecutan en contexto de usuario no contexto de kernel
  - Conocidos como servicios, subsistemas, demonios

- Programas de aplicación
  - No pertenecen al sistema
  - Ejecutados por los usuarios
  - No suele considerarse parte del SO
  - Lanzado por la línea de comando, clic del ratón.

Diseño e implementación del Sistema Operativo

- Diseño e implementación de OS no existen soluciones completas, pero algunos enfoques han demostrado ser exitosos.
- La estructura interna de los diferentes sistemas operativos puede variar ampliamente.
- Iniciar el diseño mediante la definición de objetivos y especificaciones.
- Afectado por la elección del hardware, tipo de sistema
- Objetivos del usuario y objetivos del sistema:
  - Objetivos del usuario - sistema operativo debe ser conveniente de usar, fácil de aprender, fiable, seguro y rápido.
  - Objetivos del sistema: el sistema operativo debe ser fácil de diseñar, implementar y mantener, así como flexible, confiable, libre de errores y eficiente.

Diseño e implementación del Sistema Operativo

- Principio importante para separar
  - Política: ¿Qué se hará?
  - Mecanismo: ¿Cómo hacerlo?
- Los mecanismos determinan cómo hacer algo, las políticas deciden qué se hará.
- La separación de la política de los mecanismos es un principio muy importante, que permite una máxima flexibilidad si las decisiones de política se cambian más tarde.
- Especificar y diseñar un sistema operativo es una tarea altamente creativa de la ingeniería de software.

Implementación

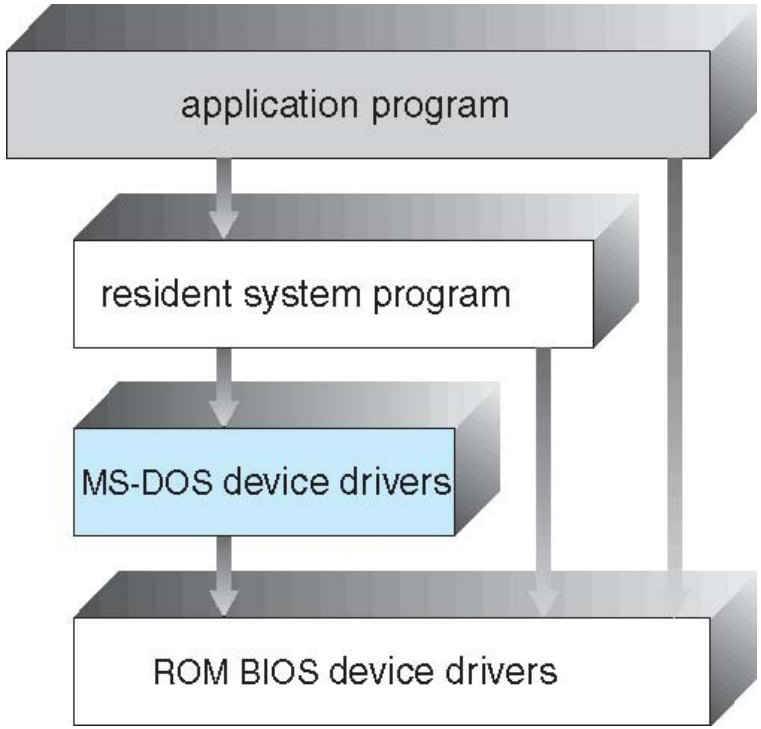
- Mucha variación
  - Sistemas operativos iniciales en asembler
  - Después se usuaron para la programación de sistema los lenguajes Algol, PL/1
  - Ahora C, C ++
- En la actualidad usualmente una mezcla de idiomas
  - Los niveles más bajos en assembler
  - Cuerpo principal en C
  - Programas de sistemas en C, C ++, lenguajes de scripting como PERL, Python, scripts de shell.
- Más alto nivel de idioma más fácil de portar a otro hardware
  - Pero más lento
- La emulación puede permitir que un sistema operativo se ejecute en hardware no nativo.

Estructura Sistema Operativo

- OS de uso general es un programa muy grande
- Varias formas de estructurarlas
  - Estructura simple - MS-DOS
  - Más complejo - UNIX
  - En capas - una abstracción
  - Microkernel -Mach

Estructura Simple - MS-DOS

- MS-DOS - escrito para proporcionar la mayor funcionalidad en el menor espacio.
  - No dividido en módulos
  - Aunque MS-DOS tiene alguna estructura, sus interfaces y niveles de funcionalidad no están bien separados.



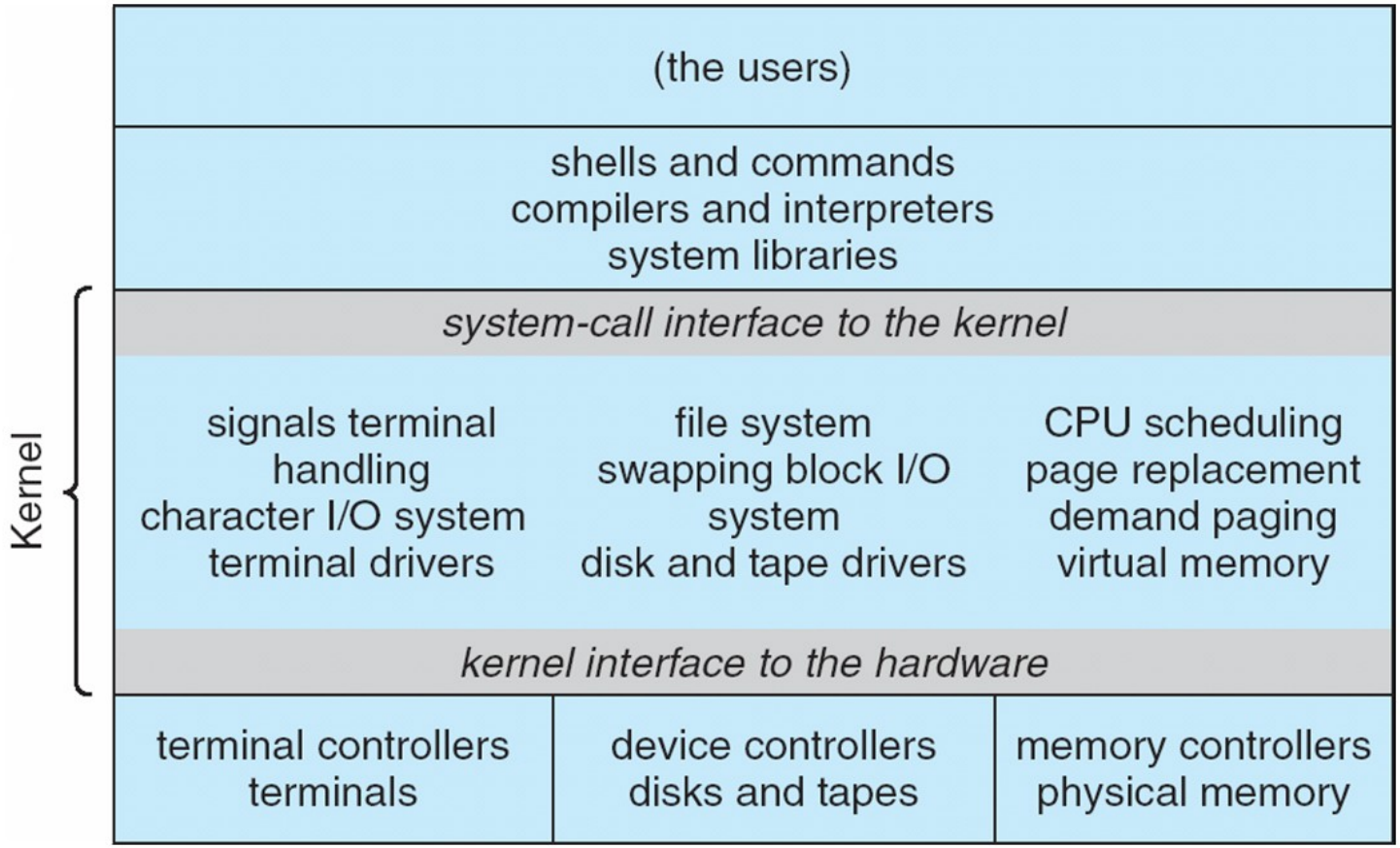
Estructura No Simple -- UNIX

UNIX - limitado por la funcionalidad de hardware, el sistema operativo UNIX original tenía estructuración limitada. El sistema operativo UNIX consta de dos partes separables

- Programas de sistemas
- El núcleo
  - Consiste en todo debajo de la interfaz de llamada al sistema y por encima del hardware físico.
  - Proporciona el sistema de archivos, programación de CPU, administración de memoria y otras funciones del sistema operativo; Un gran número de funciones para un nivel.

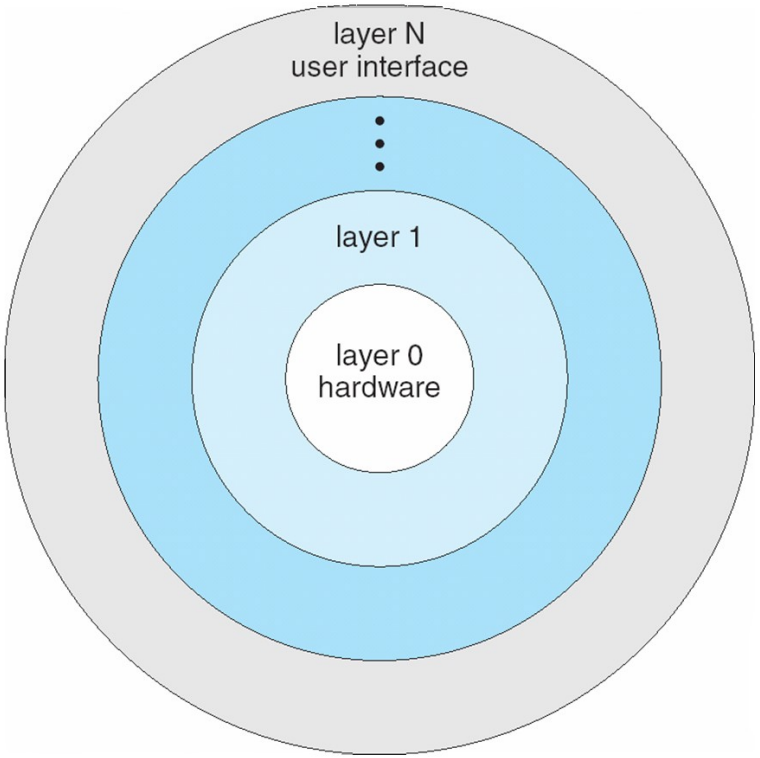
Estructura de sistema UNIX tradicional

Simple, pero no totalmente en separado en capas



Enfoque en capas

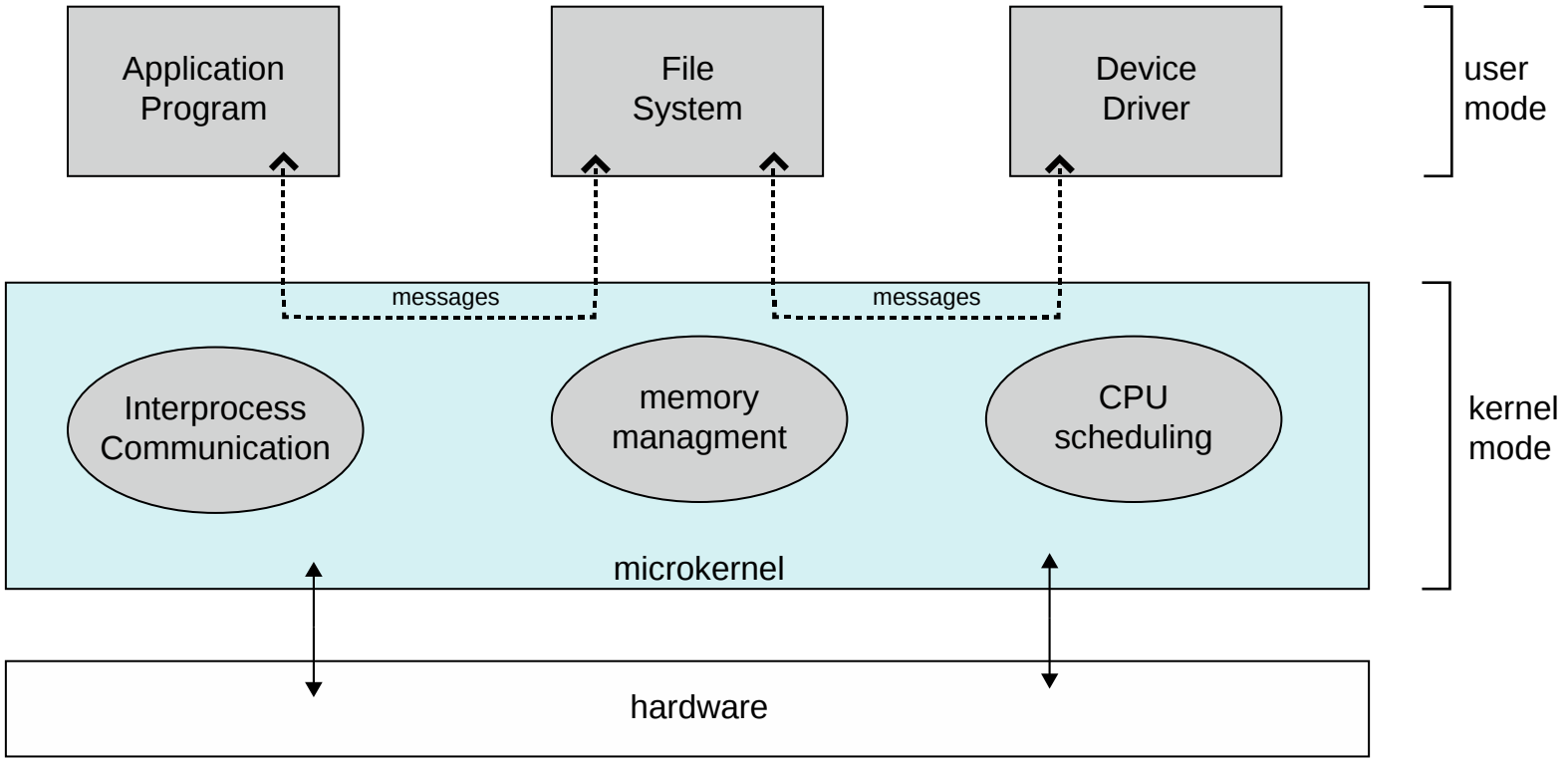
- El sistema operativo está dividido en varias capas (niveles), cada una construida sobre las capas inferiores. La capa inferior (capa 0), es el hardware; La más alta (capa N) es la interfaz de usuario.
- Con la modularidad, las capas se seleccionan de forma que cada una utiliza funciones (operaciones) y servicios de sólo capas de nivel inferior



Estructura Sistema Microkernel

- Se mueve tanto desde el kernel hacia el espacio de usuario
- Mach Ejemplo de microkernel
  - Mac OS X kernel (Darwin) en parte está basado en Mach
- La comunicación tiene lugar entre los módulos de usuario mediante el paso de mensajes
- Beneficios:
  - Más fácil de extender un microkernel
  - Más fácil de portar el sistema operativo a nuevas arquitecturas
  - Más fiable (menos código se ejecuta en modo kernel)
  - Más seguro
- Perjuicios:
  - Sobrecarga de rendimiento del espacio de usuario en la comunicación espacial del núcleo

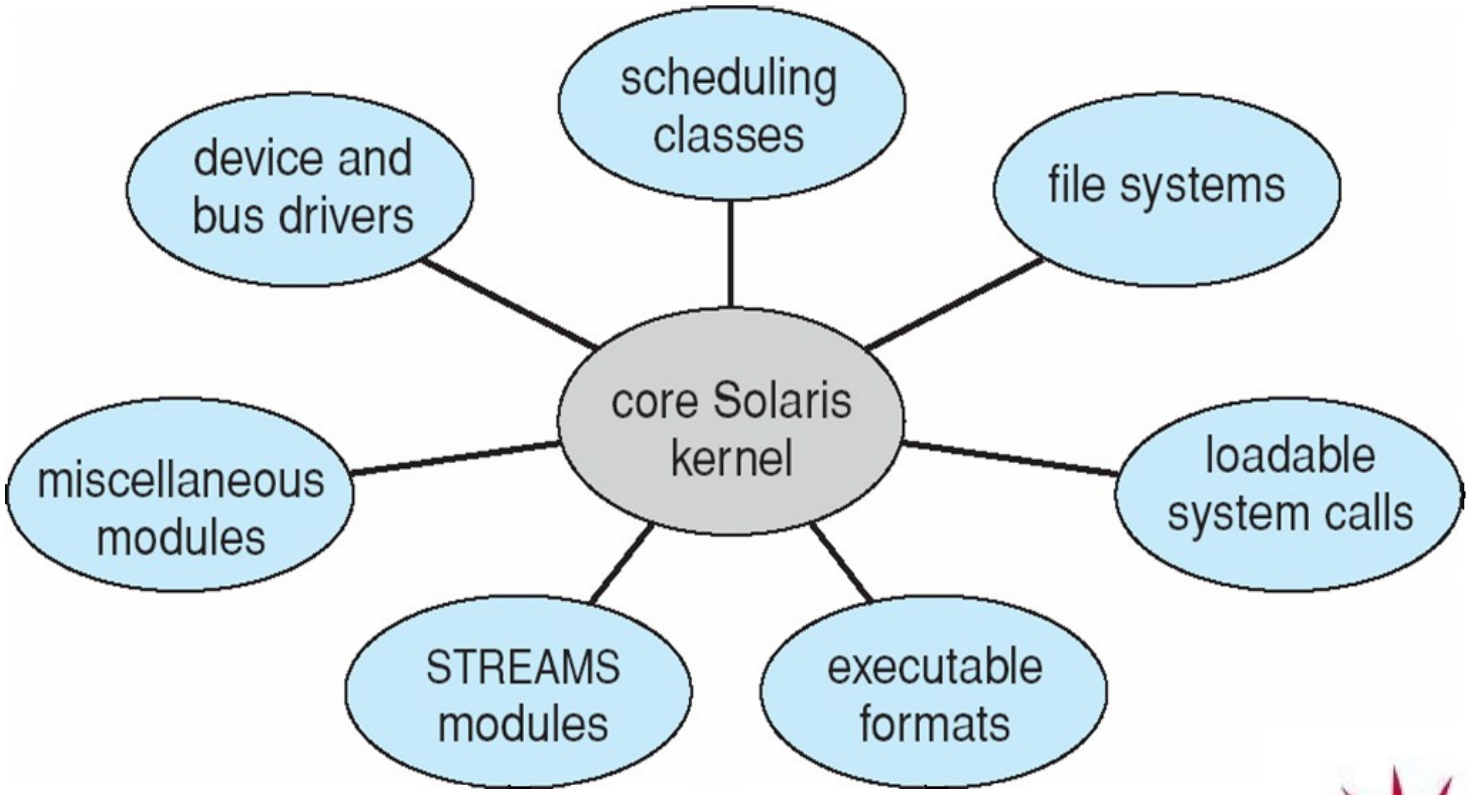
Estructura Sistema Microkernel



Módulos

- Muchos sistemas operativos modernos implementan módulos cargables del kernel
  - Utiliza un enfoque orientado a objetos
  - Cada componente central está separado
  - Cada uno habla con los demás sobre interfaces conocidas
  - Cada uno es cargable según sea necesario dentro del núcleo
- En general, similar a las capas, pero con más flexibilidad
  - Linux, Solaris, etc

Enfoque modular de Solaris

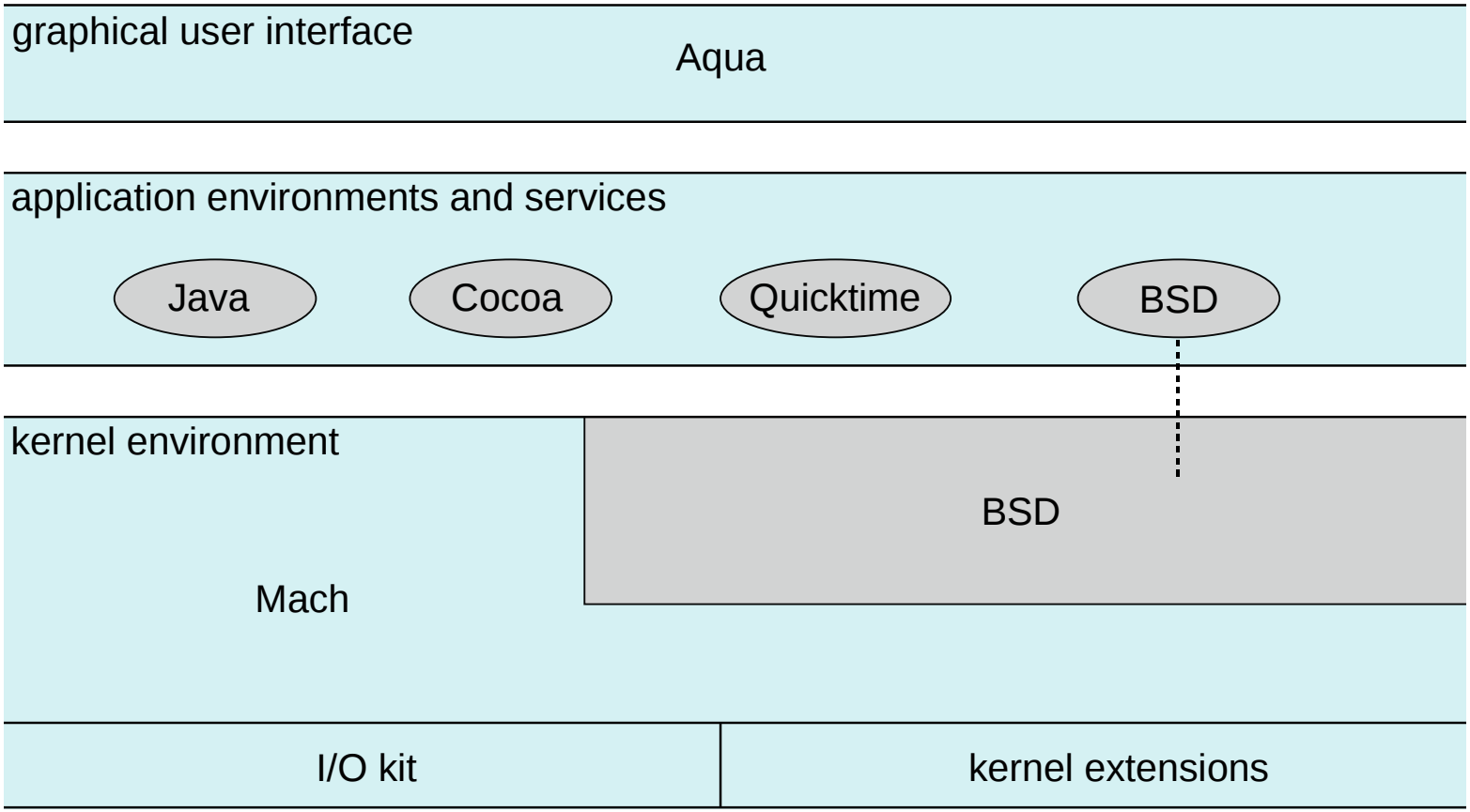


SOLARIS™

Sistemas Híbridos

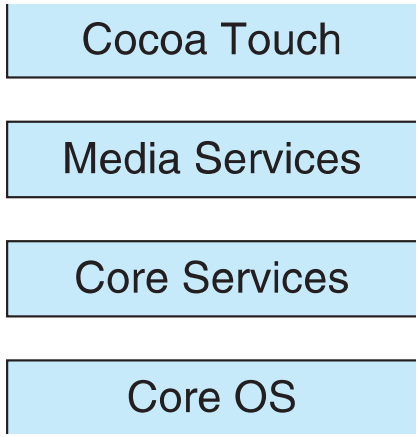
- La mayoría de los sistemas operativos modernos no son realmente un modelo puro
  - Híbridos combina múltiples enfoques para abordar el rendimiento, la seguridad, las necesidades de usabilidad
  - Los kernels Linux y Solaris en el espacio de direcciones del kernel, son monolítico, y además de modular para la carga dinámica de las funcionalidades.
  - Windows principalmente monolítico, además de microkernel para diferentes subsistema personalizables.
- Apple Mac OS X híbrido, capas, Aqua UI más Cocoa son el entorno de programación
  - A continuación se muestra el núcleo que consta de microkernel Mach y BSD Unix, además de kit de I/O y módulos dinámicamente cargables (llamados extensiones del kernel)

Estructura Mac OS X



iOS

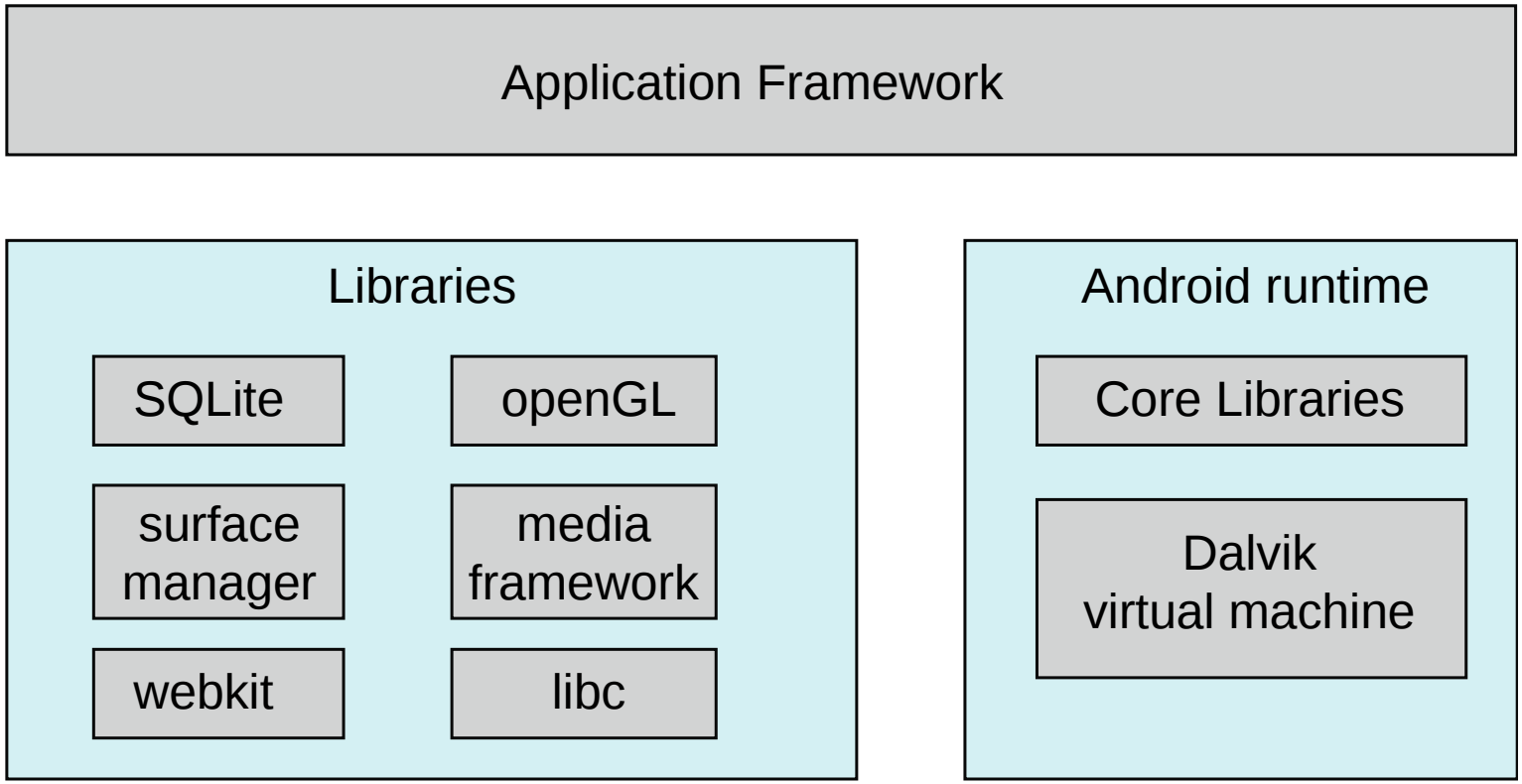
- Sistema operativo móvil de Apple para iPhone, iPad
  - Estructurado en Mac OS X, funcionalidad añadida.
  - No ejecuta aplicaciones de OS X de forma nativa
    - También se ejecuta en la arquitectura de CPU diferente (ARM vs Intel).
  - API Cocoa Touch Objective-C para desarrollar aplicaciones.
  - Capa de servicios multimedia para gráficos, audio y video
  - Core Services proporciona cloud computing, bases de datos.
  - Nucleo Sistema operativo, basado en el núcleo de Mac OS X



Android

- Desarrollado por Open Handset Alliance (principalmente Google)
  - Código abierto (Open Source)
- Similar al stack IOS
- Basado en el kernel de Linux, pero modificado
  - Proporciona gestión de procesos, memoria, controladores de dispositivo
  - Añade administración de energía
- El entorno de tiempo de ejecución incluye un conjunto básico de bibliotecas y la máquina virtual Dalvik
  - Aplicaciones desarrolladas en Java y en la API de Android
    - Los archivos de clase Java compilados a bytecode de Java luego traducidos a ejecutable que ejecuta en VM de Dalvik
- Las bibliotecas incluyen marcos para el navegador web (webkit), base de datos (SQLite), multimedia, libc más pequeño.

Arquitectura Android

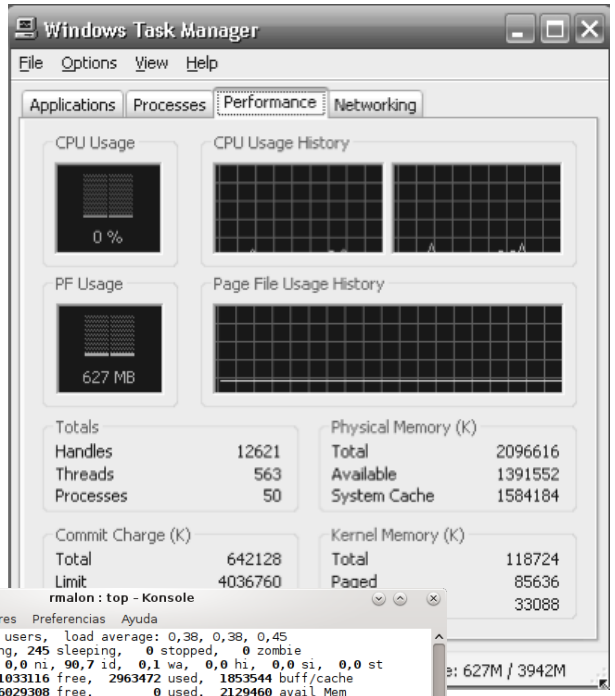


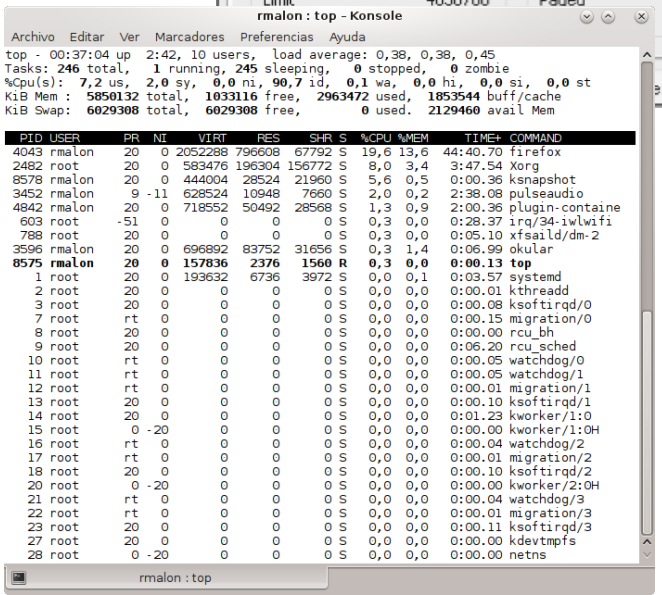
Depuración del Sistema Operativo

- La depuración permite encontrar y corregir errores (bugs)
- OS generan archivos de registro que contienen información de error.
- El error de una aplicación puede generar archivos de volcado de núcleo que capturan la memoria del proceso.
- El fallo del sistema operativo puede generar un archivo de volcado de bloqueo que contenga memoria del núcleo
- Más allá de fallos, el ajuste de rendimiento puede optimizar el rendimiento del sistema

La optimización del rendimiento

- Mejorar el rendimiento al eliminar los cuellos de botella.
- OS debe proporcionar medios de computación y mostrar medidas de comportamiento del sistema
- Por ejemplo, "top" de Linux/Unix o el Administrador de tareas de Windows





System Boot

- Cuando la energía se inicializa en el sistema, la ejecución comienza en una ubicación de memoria fija
  - ROM de firmware utilizada para mantener el código inicial de arranque
- El sistema operativo debe estar disponible para que el hardware lo pueda iniciar
  - Pequeño trozo de código - cargador de arranque, almacenado en ROM o EEPROM localiza el kernel, lo carga en la memoria y lo inicia
  - A veces proceso de dos pasos donde el bloque de arranque en una ubicación fija cargado por código ROM, que carga el cargador de arranque desde el disco
- GRUB (cargador de arranque), permite la selección de kernel desde varios discos, versiones, opciones de kernel
- El kernel se carga y el sistema entonces se ejecuta