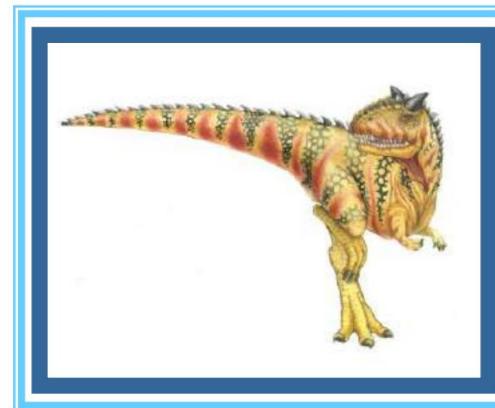


Capítulo 2: Sistema operativo Servicios





Describir

- Servicios del sistema operativo
- Interfaz de usuario y sistema operativo
- Llamadas al sistema ▪ Servicios del sistema
- Enlazadores y cargadores
- Por qué las aplicaciones son específicas del sistema operativo
- **Diseño e Implementación**
- Estructura del sistema operativo
- Creación e inicio de un sistema operativo ▪ Depuración del sistema operativo

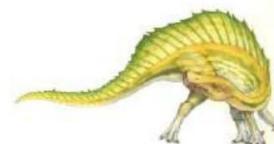




Objetivos

- Identificar los servicios proporcionados por un sistema operativo.
- Ilustrar cómo se utilizan las llamadas al sistema para proporcionar servicios del sistema operativo.
- Comparar y contrastar estrategias monolíticas, en capas, de micronúcleo, modulares e híbridas para diseñar sistemas operativos.

- Ilustrar el proceso para iniciar un sistema operativo
- Aplicar herramientas para monitorear el desempeño del sistema operativo.
- Diseñar e implementar módulos del kernel para interactuar con un núcleo de linux

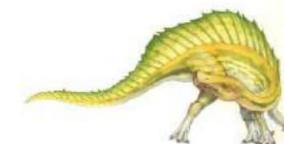




Servicios del sistema operativo

- Los sistemas operativos proporcionan un entorno para la ejecución de programas.
y 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 [línea de comandos \(CLI\)](#), [usuario de gráficos Interfaz \(GUI\)](#), pantalla táctil, lote
 - Ejecución del programa : el sistema debe poder cargar un programa en la memoria y, para ejecutarlo, finalizar la ejecución, ya sea de forma normal o anormal (lo que indica un error).
 - Operaciones de E/S : un programa en ejecución puede requerir E/S, lo que puede implicar un archivo o un dispositivo de E/S.
 - Manipulación del sistema de archivos : el sistema de archivos es de particular importancia. interés. Los programas necesitan leer y escribir archivos y directorios, crearlos y eliminarlos, buscarlos, enumerar información de archivos y administrar permisos.





Servicios del sistema operativo (cont.)

- Un conjunto de servicios del sistema operativo proporciona funciones que son útiles para el usuario (Cont.):
 - 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 paso de mensajes (paquetes movidos por el SO)
 - Detección de errores : el sistema operativo debe estar constantemente al tanto de posibles errores

Puede ocurrir en la CPU y el hardware de memoria, en dispositivos de E/S, en programa de usuario

Para cada tipo de error, el sistema operativo debe tomar las medidas adecuadas para garantizar una computación correcta y consistente

Las funciones de depuración pueden mejorar en gran medida las capacidades del usuario y del programador para utilizar eficientemente el sistema.





Servicios del sistema operativo (cont.)

- Existe otro conjunto de funciones del sistema operativo para garantizar el funcionamiento eficiente del propio sistema a través del intercambio de recursos
 - Asignación de recursos: cuando hay varios usuarios o varios trabajos ejecutándose simultáneamente, se deben asignar recursos a cada uno de ellos

Muchos tipos de recursos: ciclos de CPU, memoria principal, archivos almacenamiento, dispositivos de E/S.
 - Registro: para realizar un seguimiento de qué usuarios utilizan cuánto y qué tipo de recursos informáticos
 - Protección y seguridad: los propietarios de la 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.

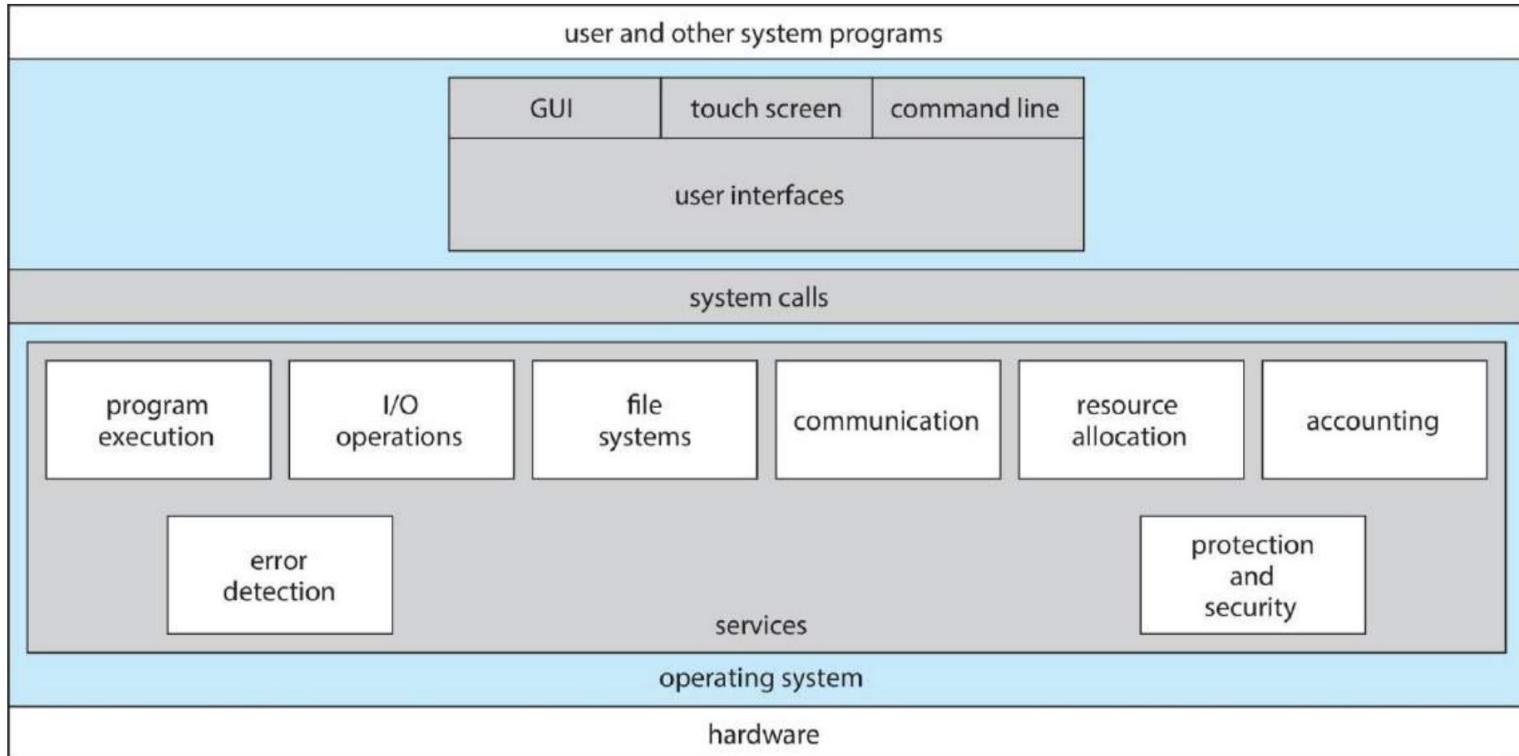
La protección implica garantizar que todo el acceso a los recursos del sistema esté controlado

La seguridad del sistema frente a personas externas requiere autenticación del usuario y se extiende a la defensa de dispositivos de E/S externos contra intentos de acceso no válidos.





Una vista de los servicios del sistema operativo

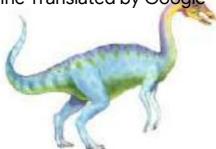




Intérprete de línea de comando

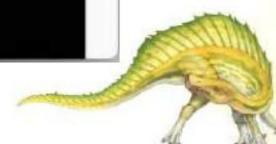
- CLI permite la entrada directa de comandos
- A veces implementado en el kernel, a veces mediante programa del sistema.
- A veces se implementan múltiples sabores: **shells**
- Principalmente obtiene un comando del usuario y lo ejecuta
- A veces, comandos integrados, a veces solo nombres de programas.
 - Si es lo último, agregar nuevas funciones no requiere modificación del caparazón

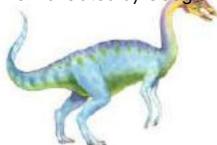




Intérprete de comandos de Bourne Shell

```
1. root@r6181-d5-us01:~ (ssh)
X root@r6181-d5-u... ● %1 X ssh %2 X root@r6181-d5-us01... %3
Last login: Thu Jul 14 08:47:01 on ttys002
iMacPro:~ pbg$ ssh root@r6181-d5-us01
root@r6181-d5-us01's password:
Last login: Thu Jul 14 06:01:11 2016 from 172.16.16.162
[root@r6181-d5-us01 ~]# uptime
06:57:48 up 16 days, 10:52, 3 users, load average: 129.52, 80.33, 56.55
[root@r6181-d5-us01 ~]# df -kh
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/vg_ks-lv_root
                      50G   19G   28G  41% /
tmpfs                 127G  520K  127G   1% /dev/shm
/dev/sda1              477M   71M  381M  16% /boot
/dev/dssd0000          1.0T  480G  545G  47% /dssd_xfs
tcp://192.168.150.1:3334/orangefs
                      12T  5.7T  6.4T  47% /mnt/orangefs
/dev/gpfs-test         23T  1.1T  22T   5% /mnt/gpfs
[root@r6181-d5-us01 ~]#
[root@r6181-d5-us01 ~]# ps aux | sort -nrk 3,3 | head -n 5
root      97653 11.2  6.6 42665344 17520636 ?  S<Ll Jul13 166:23 /usr/lpp/mmfs/bin/mmfsd
root      69849  6.6  0.0     0     0 ?        S    Jul12 181:54 [vpthread-1-1]
root      69850  6.4  0.0     0     0 ?        S    Jul12 177:42 [vpthread-1-2]
root      3829  3.0  0.0     0     0 ?        S   Jun27 730:04 [rp_thread 7:0]
root      3826  3.0  0.0     0     0 ?        S   Jun27 728:08 [rp_thread 6:0]
[root@r6181-d5-us01 ~]# ls -l /usr/lpp/mmfs/bin/mmfsd
-r-x----- 1 root root 20667161 Jun  3  2015 /usr/lpp/mmfs/bin/mmfsd
[root@r6181-d5-us01 ~]#
```





Interfaz del sistema operativo de usuario: GUI

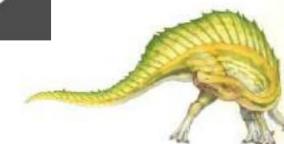
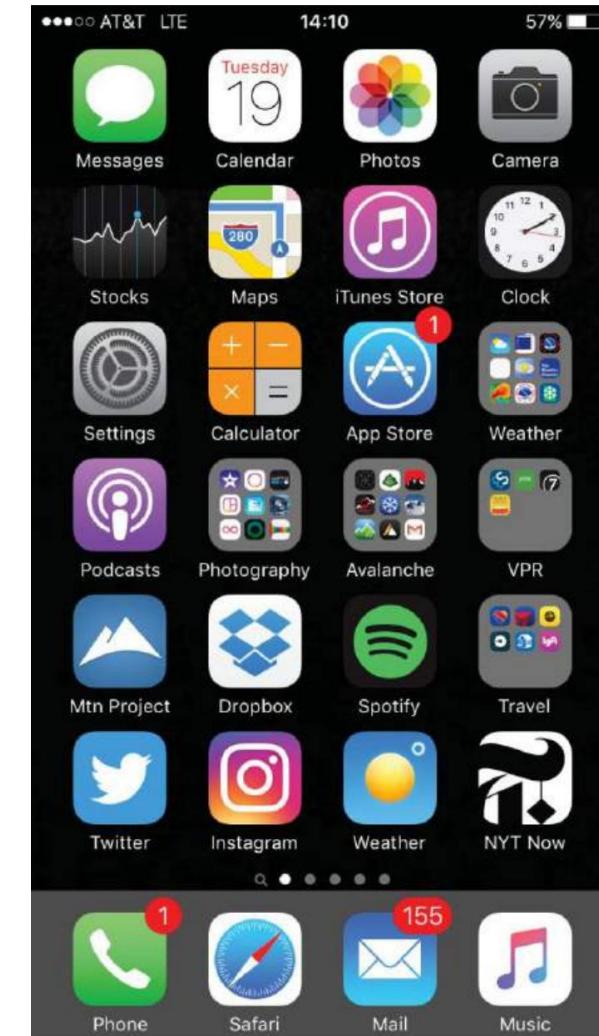
- Interfaz de metáfora **de escritorio** fácil de usar
 - Generalmente mouse, teclado y monitor
 - **Los iconos** representan archivos, programas, acciones, etc.
 - Varios botones del mouse sobre objetos en la interfaz provocan diversas acciones (proporcionar información, opciones, ejecutar funciones, abrir directorio (conocido como **carpeta**)
 - Inventado en Xerox PARC
- Muchos sistemas ahora incluyen interfaces CLI y GUI
 - Microsoft Windows es GUI con shell de “comando” CLI
 - Apple Mac OS X es una interfaz GUI “Aqua” con kernel UNIX debajo y conchas disponibles
 - Unix y Linux tienen CLI con interfaces GUI opcionales (CDE, KDE, GNOMO)





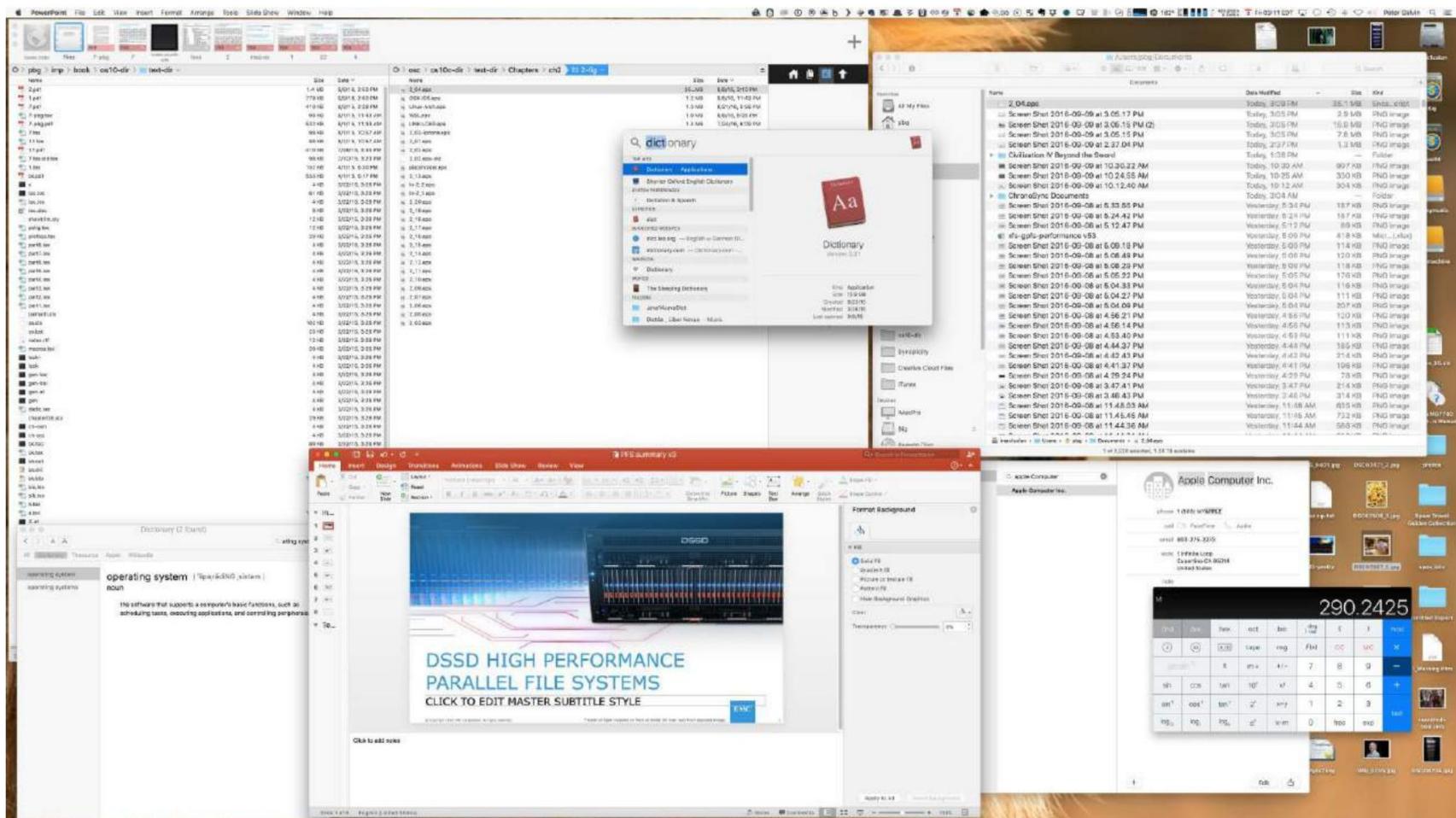
Interfaces de pantalla táctil

- Los dispositivos con pantalla táctil requieren nuevas interfaces
 - El mouse no es posible o no es deseado
 - Acciones y selección basada en gestos
 - Teclado virtual para entrada de texto
- Comandos de voz





La interfaz gráfica de usuario de Mac OS X





Llamadas al sistema

- Interfaz de programación para los servicios proporcionados por el SO.
- Normalmente escrito en un lenguaje de alto nivel (C o C++)
- Los programas acceden principalmente a través de una [aplicación](#) de alto nivel.
[Interfaz de programación \(API\)](#) en lugar de uso directo de llamadas al sistema
- Las tres API más comunes son la API Win32 para Windows, la API POSIX para Sistemas basados en POSIX (incluidas prácticamente todas las versiones de UNIX, Linux y Mac OS X) y API de Java para la máquina virtual Java (JVM)

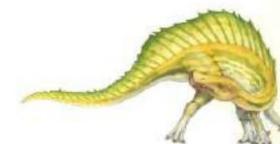
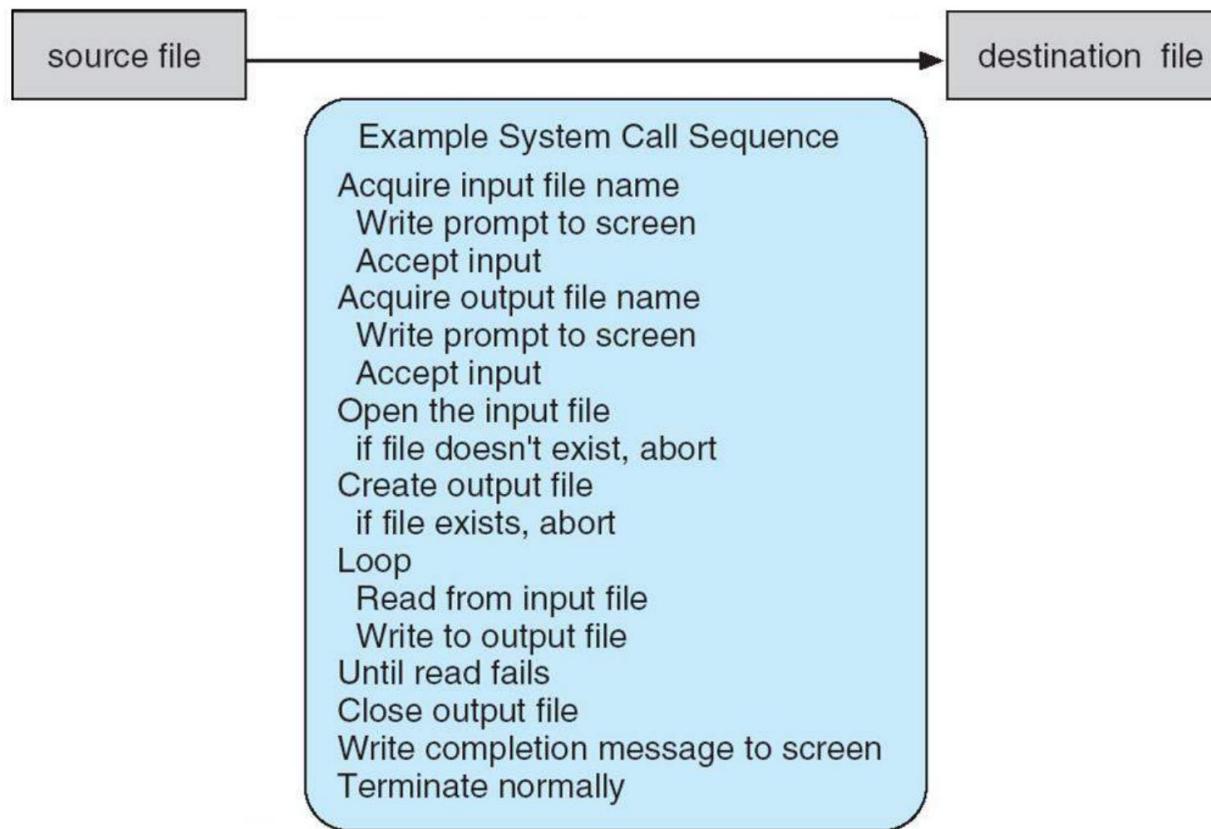
Tenga en cuenta que los nombres de llamadas al sistema utilizados a lo largo de este texto son genéricos.

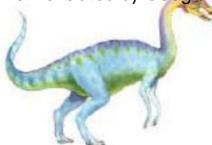




Ejemplo de llamadas al sistema

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





Ejemplo de API estándar

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 function parameters
value name

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 into which the data will be read
- `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 de llamadas al sistema

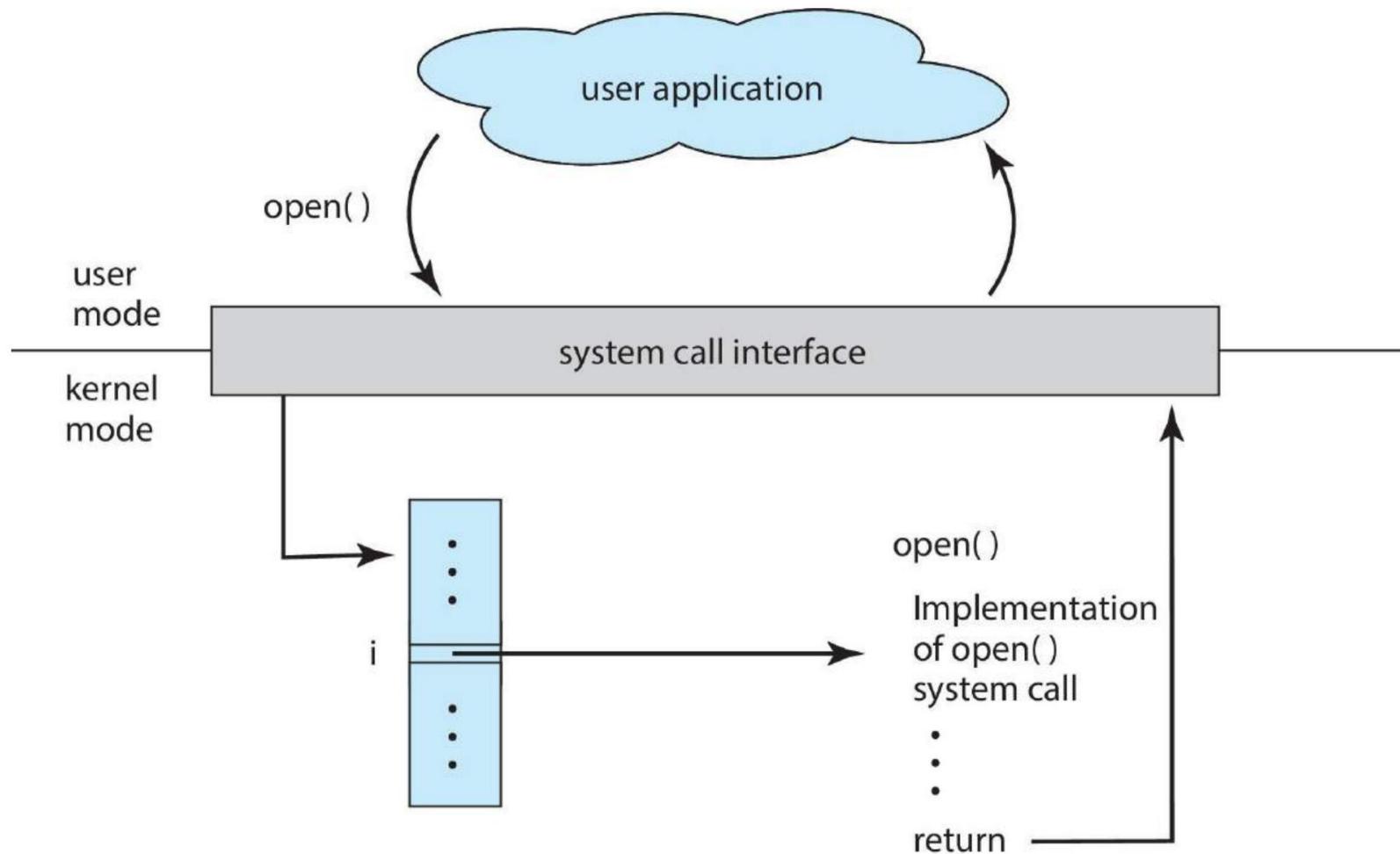
- Normalmente, se asocia un número con cada llamada al sistema.
 - La interfaz de llamada al sistema mantiene una tabla indexada según estos números
- La interfaz de llamada al sistema invoca la llamada al sistema prevista en el kernel del sistema operativo y devuelve el estado de la llamada al sistema y cualquier valor de retorno.
- La persona que llama no necesita saber nada sobre cómo se implementa la llamada al sistema.
 - Sólo necesita obedecer la API y comprender qué hará el sistema operativo como llamada de resultado
 - La API oculta la mayoría de los detalles de la interfaz del sistema operativo al programador.

Administrado por la biblioteca de soporte en tiempo de ejecución (conjunto de funciones integradas en las bibliotecas incluidas con el compilador)





API – Llamada al sistema – Relación con el sistema operativo





Paso de parámetros de llamada al sistema

- A menudo, se requiere más información que simplemente la identidad del producto deseado.

llamada al sistema

- El tipo exacto y la cantidad de información varían según el sistema operativo y

llamar

- Tres métodos generales utilizados para pasar parámetros al sistema operativo

- Más simple: pasar los parámetros en registros

En algunos casos, puede haber más parámetros que registros.

- Parámetros almacenados en un bloque o tabla en la memoria y dirección del bloque pasado como parámetro en un registro.

Este enfoque adoptado por Linux y Solaris

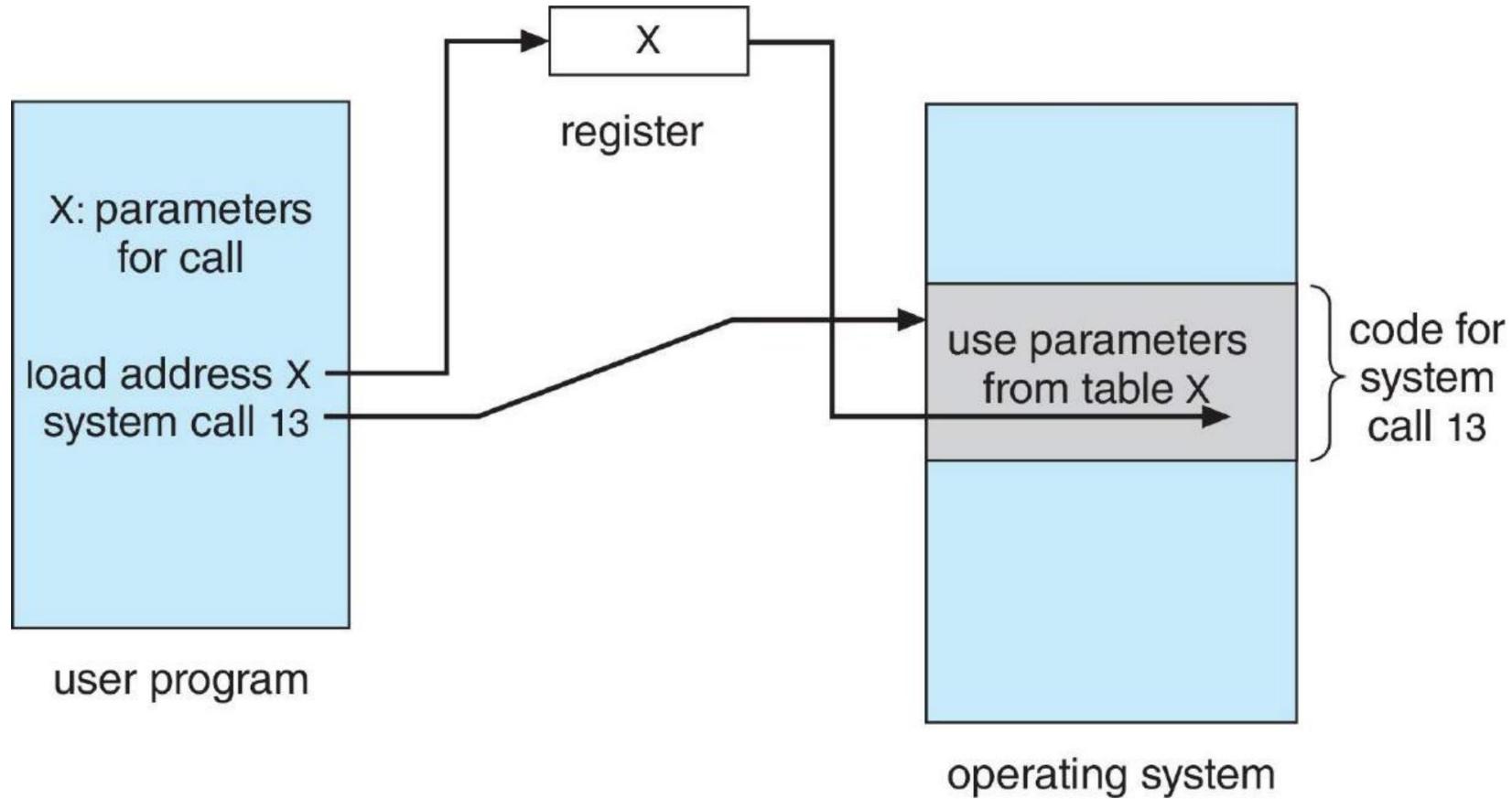
- Parámetros colocados o **empujados** a la **pila** por el programa y **sacado** de la pila por el sistema operativo

- Los métodos de bloque y pila no limitan el número o la longitud de los parámetros que se pasan





Paso de parámetros a través de la tabla





Tipos de llamadas al sistema

- Control de procesos
 - crear proceso, terminar proceso
 - finalizar, abortar
 - cargar, ejecutar
 - obtener atributos de proceso, establecer atributos de proceso
 - esperar el tiempo
 - esperar evento, señalar evento
 - asignar y liberar memoria
 - Volcar memoria si se produce un error
- **Depurador** para determinar errores, ejecución en un solo paso
- **Bloqueos** para gestionar el acceso a datos compartidos entre procesos





Tipos de llamadas al sistema (cont.)

- Gestión de archivos • crear

- archivo, eliminar archivo

- abrir, cerrar archivo •

- leer, escribir, reposicionar •

- obtener y establecer atributos de archivo

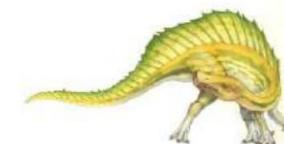
- Gestión de dispositivos

- solicitar dispositivo, liberar dispositivo

- leer, escribir, reposicionar •

- obtener atributos del dispositivo, establecer atributos del

- dispositivo • conectar o desconectar dispositivos lógicamente





Tipos de llamadas al sistema (cont.)

- Mantenimiento de la información
 - obtener hora o fecha, establecer hora o fecha
 - obtener datos del sistema, configurar datos del sistema
 - obtener y configurar atributos de proceso, archivo o dispositivo
- Comunicaciones
 - crear, eliminar conexión de comunicación
 - enviar y recibir mensajes si **el modelo de transmisión de mensajes al host**
nombre o nombre del proceso
Del cliente al servidor
 - **El modelo de memoria compartida** crea y obtiene acceso a la memoria.
regiones
 - información del estado de la transferencia
 - conectar y desconectar dispositivos remotos





Tipos de llamadas al sistema (cont.)

- Protección
 - Controlar el acceso a los recursos
 - Obtener y establecer permisos
 - Permitir y denegar el acceso de los usuarios





Ejemplos de llamadas al sistema Windows y Unix

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

The following illustrates various equivalent system calls for Windows and UNIX operating systems.

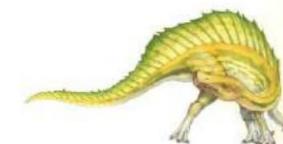
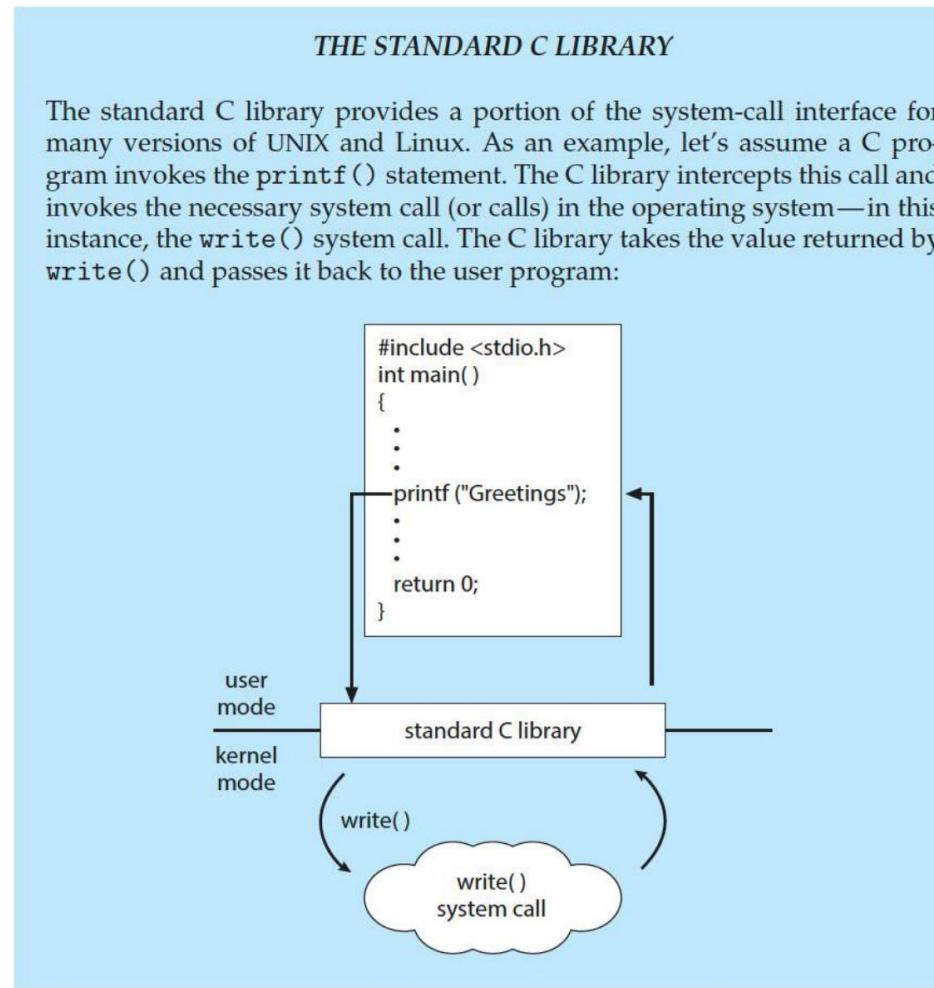
	Windows	Unix
Process control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communications	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shm_open() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()





Ejemplo de biblioteca C estándar

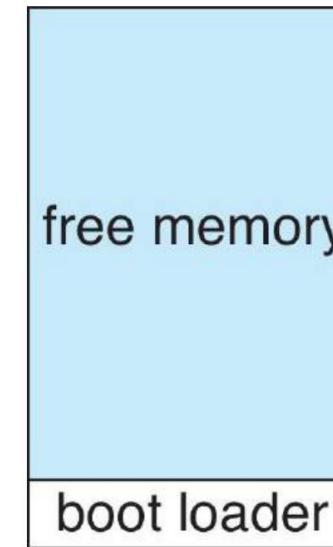
- Programa C que invoca la llamada a la biblioteca printf(), que llama a la llamada al sistema write()





Ejemplo: Arduino

- Tarea única
- Sin sistema operativo
- Programas (sketch) cargados vía USB a memoria flash
- Espacio de memoria único
- El cargador de arranque carga el programa
- Salida del programa -> shell recargado



(a)

Al iniciar el sistema



(b)

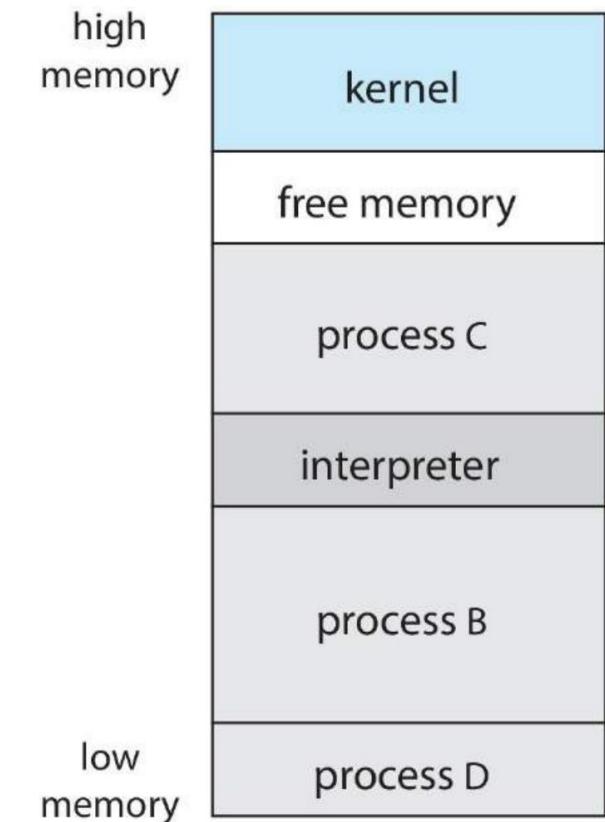
ejecutando un programa





Ejemplo: FreeBSD

- Variante Unix
- Multitarea
- Inicio de sesión de usuario -> invocar la elección del usuario
caparazón
- Shell ejecuta la llamada al sistema fork() para crear el proceso.
 - Ejecuta exec() para cargar el programa en proceso
 - Shell espera a que finalice el proceso o continúa con los comandos del usuario
- El proceso sale con:
 - código = 0 – sin error
 - código > 0 – código de error





Servicios del sistema

- Los programas del sistema proporcionan un entorno conveniente para el desarrollo y ejecución de programas. Se pueden dividir en:
 - Manipulación de archivos
 - Información de estado a veces almacenada en un archivo
 - Soporte de lenguaje de programación
 - **Carga y ejecución del programa**
 - Comunicaciones
 - Servicios de fondo
 - Programas de aplicación
- La visión de la mayoría de los usuarios sobre el sistema operativo está definida por el sistema.
programas, no las llamadas reales al sistema





Servicios del sistema (cont.)

- Proporcionar un entorno conveniente para el desarrollo de programas y ejecución
 - Algunos de ellos son simplemente interfaces de usuario para llamadas al sistema; otros son considerablemente más complejos
- Gestión de archivos : crear, eliminar, copiar, cambiar el nombre, imprimir, volcar, enumerar, y en general manipular archivos y directorios
- Información de estado
 - Algunos solicitan información al sistema: fecha, hora, cantidad de memoria disponible, espacio en disco, número de usuarios.
 - Otros proporcionan rendimiento, registro y depuración detallados. información
 - Normalmente, estos programas formatean e imprimen la salida en el terminal u otros dispositivos de salida
 - Algunos sistemas implementan un **registro** , que se utiliza para almacenar y recuperar información de configuración





Servicios del sistema (cont.)

- Modificación de archivos
 - Editores de texto para crear y modificar archivos.
 - Comandos especiales para buscar contenidos de archivos o realizar transformaciones del texto
- Compatibilidad con lenguajes de programación : en ocasiones se proporcionan compiladores, ensambladores, depuradores e intérpretes.
- Carga y ejecución de programas : cargadores absolutos, reubicables cargadores, editores de enlaces y cargadores de superposiciones, sistemas de depuración para lenguaje de máquina y de nivel superior
- Comunicaciones : proporciona el mecanismo para crear conexiones virtuales entre procesos, usuarios y sistemas informáticos.
 - Permitir a los usuarios enviar mensajes a las pantallas de otros, navegar por páginas web, enviar mensajes de correo electrónico, iniciar sesión de forma remota y transferir archivos de una máquina a otra.





Servicios del sistema (cont.)

▪ Servicios de antecedentes

- Iniciar en el momento del arranque

Algunos para iniciar el sistema y luego finalizar

Algunos desde el inicio del sistema hasta el apagado

- Proporcionar servicios como comprobación de disco, programación de procesos, registro de errores e impresión.
- Ejecutar en el contexto del usuario, no en el contexto del kernel.
- Conocidos como [servicios](#), [subsistemas](#), [demonios](#).

▪ Programas de aplicación

• No pertenecen al sistema

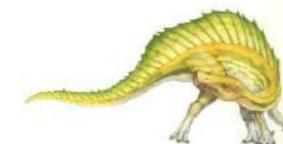
- Dirigido por usuarios
- Normalmente no se considera parte del sistema operativo.
- Se inicia mediante línea de comando, clic del mouse o toque con el dedo.





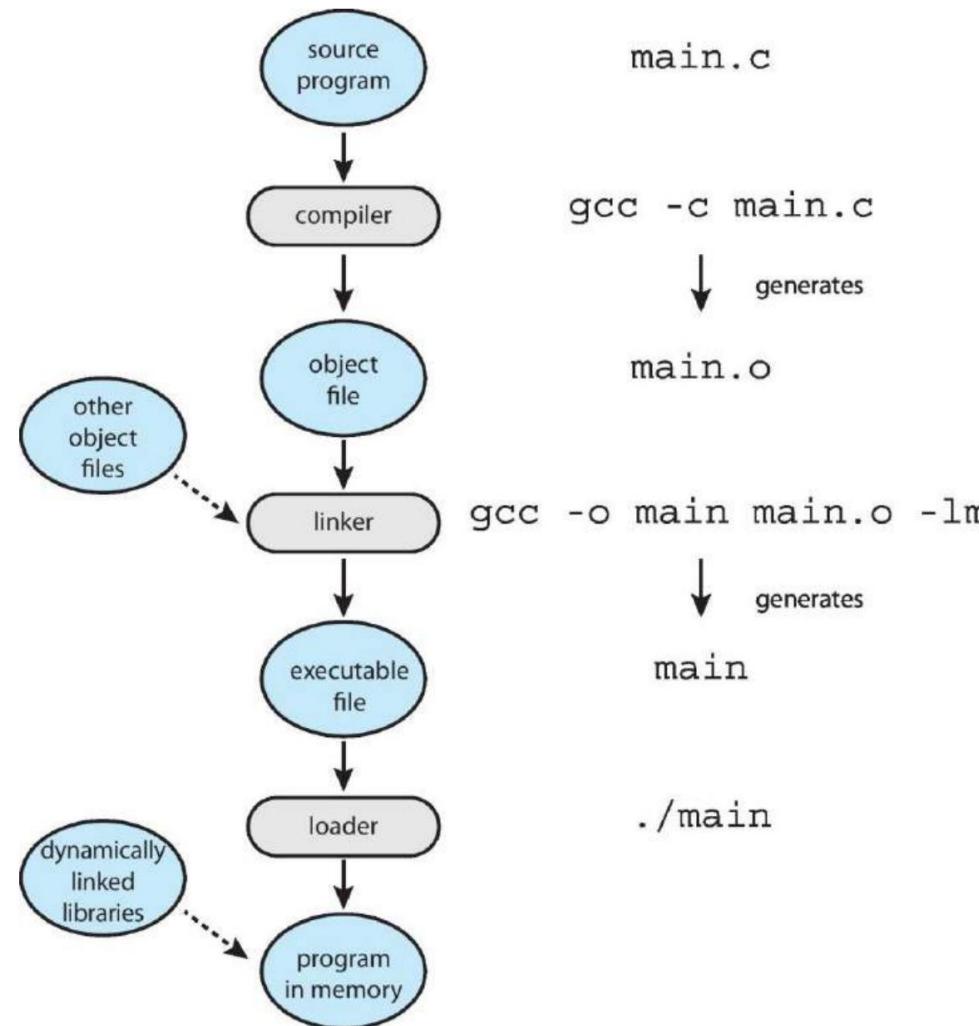
Vinculadores y cargadores

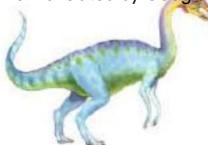
- Código fuente compilado en archivos objeto diseñados para cargarse en cualquier ubicación de memoria física: **archivo objeto reubicable**
- **El vinculador** los combina en un único archivo **ejecutable** binario.
 - También trae bibliotecas
- El programa reside en el almacenamiento secundario como ejecutable binario
- El **cargador** debe llevarlo a la memoria para ejecutarlo.
 - **La reubicación** asigna direcciones finales a las partes del programa y ajusta el código y los datos en el programa para que coincidan con esas direcciones.
- Los sistemas modernos de propósito general no vinculan bibliotecas a archivos ejecutables.
 - Más bien, **las bibliotecas vinculadas dinámicamente** (en Windows, **DLL**) son cargado según sea necesario, compartido por todos los que usan la misma versión de esa misma biblioteca (cargado una vez)
- Los archivos objeto y ejecutables tienen formatos estándar, por lo que el sistema operativo sabe cómo cargarlos e iniciarlos.





El papel del enlazador y del cargador





Por qué las aplicaciones son específicas del sistema operativo

- Las aplicaciones compiladas en un sistema generalmente no son ejecutables en otros sistemas operativos.
- Cada sistema operativo proporciona sus propias llamadas al sistema únicas
 - Formatos de archivos propios, etc.
- Las aplicaciones pueden ser multisistema operativo
 - Escrito en lenguaje interpretado como Python, Ruby e intérprete. disponible en múltiples sistemas operativos
 - Aplicación escrita en un lenguaje que incluye una máquina virtual que contiene la aplicación en ejecución (como Java)
 - Usar lenguaje estándar (como C), compilar por separado en cada sistema operativo que se ejecutará en cada
- **La interfaz binaria de aplicaciones (ABI)** es una arquitectura equivalente a la API y define cómo los diferentes componentes del código binario pueden interactuar para un sistema operativo determinado en una arquitectura, CPU, etc. determinada.





Diseño e implementación

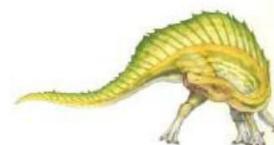
- El diseño y la implementación del sistema operativo no tiene solución, pero algunos enfoques han demostrado ser exitosos.
- La estructura interna de diferentes sistemas operativos puede variar ampliamente
- Iniciar el diseño definiendo objetivos y especificaciones.
- Afectado por la elección del hardware, tipo de sistema
- Metas del usuario y metas del sistema .
 - Objetivos del usuario: el sistema operativo debe ser cómodo de usar, fácil de aprender, confiable, seguro y rápido.
 - Objetivos del sistema: el sistema operativo debe ser fácil de diseñar, implementar y mantener, así como flexibles, confiables, libres de errores y eficientes.
- Especificar y diseñar un sistema operativo es una tarea de software altamente creativa.
ingeniería





Política y mecanismo

- Política: ¿Qué hay que hacer?
 - Ejemplo: Interrupción cada 100 segundos
- Mecanismo: ¿Cómo hacer algo?
 - Ejemplo: temporizador
- Principio importante: separar política del mecanismo
- La separación entre política y mecanismo es una cuestión muy
 - Como principio importante, permite la máxima flexibilidad si las decisiones políticas deben cambiarse más adelante.
- Ejemplo: cambiar 100 a 200





Implementación

- Mucha variación
 - Los primeros sistemas operativos en lenguaje ensamblador
 - Luego lenguajes de programación de sistemas como Algol, PL/1
 - Ahora C, C++
- En realidad suele ser una mezcla de idiomas.
 - Niveles más bajos en montaje
 - Cuerpo principal en C
 - Programas de sistemas en C, C++, lenguajes de scripting como PERL, Python, scripts de shell
- Más lenguaje de alto nivel más fácil de [migrar](#) a otro hardware
 - Pero más lento
- [La emulación](#) puede permitir que un sistema operativo se ejecute en hardware no nativo

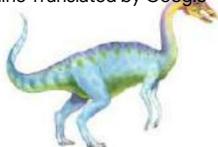




Estructura del sistema operativo

- El sistema operativo de propósito general es un programa muy grande.
- Varias formas de estructurarlos.
 - Estructura simple – MS-DOS
 - Más complejo – UNIX
 - En capas: una abstracción
 - Micronúcleo – Mach





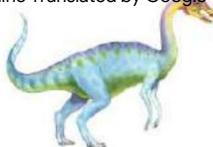
Estructura monolítica – UNIX original

- UNIX: limitado por la funcionalidad del hardware, el sistema operativo UNIX original tenía una estructura limitada.
- El sistema operativo UNIX consta de dos partes separables
 - Programas de sistemas
 - El núcleo

Consiste en todo lo que se encuentra debajo de la interfaz de llamada al sistema y encima del hardware físico.

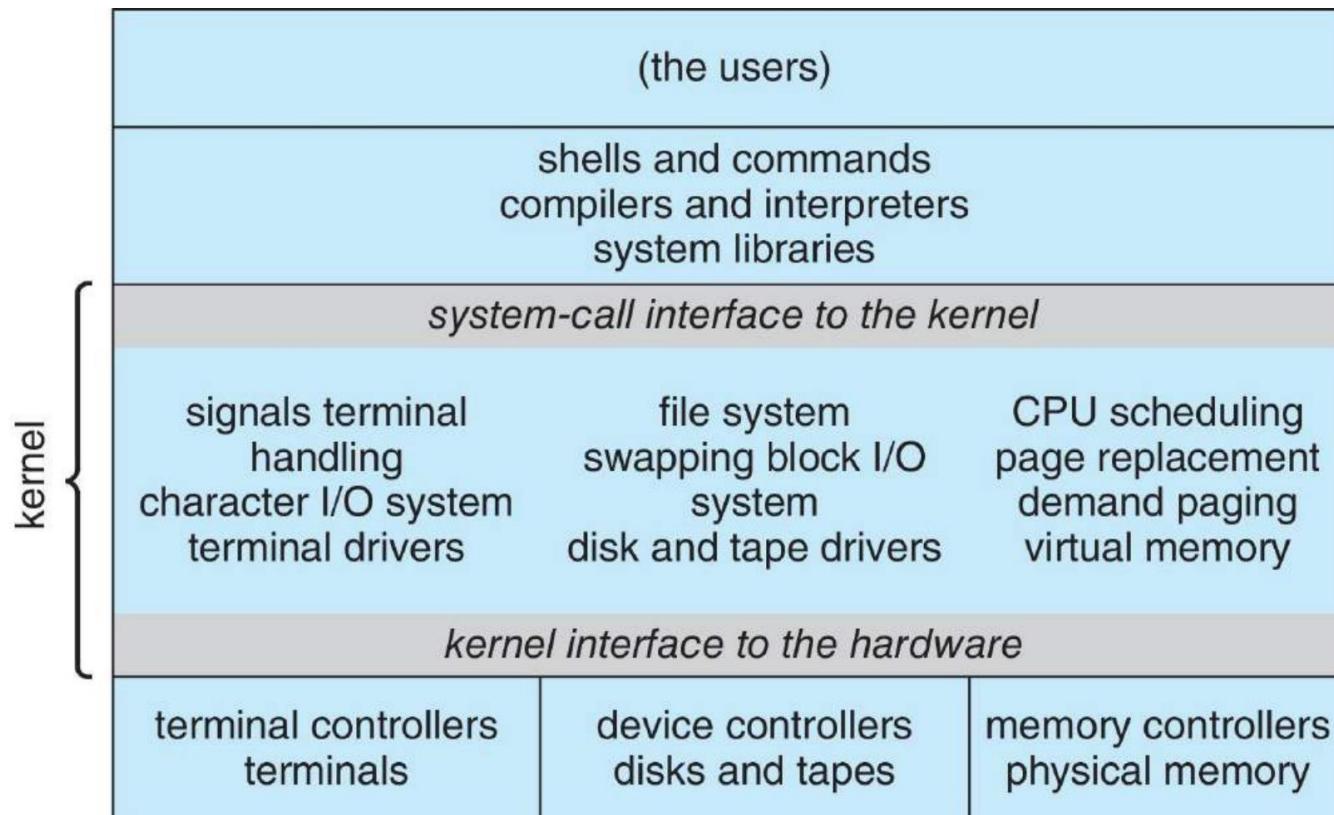
Proporciona el sistema de archivos, programación de CPU, memoria gestión y otras funciones del sistema operativo; un gran número de funciones para un nivel





Estructura del sistema UNIX tradicional

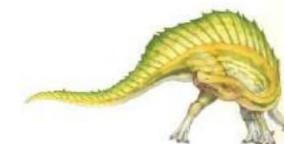
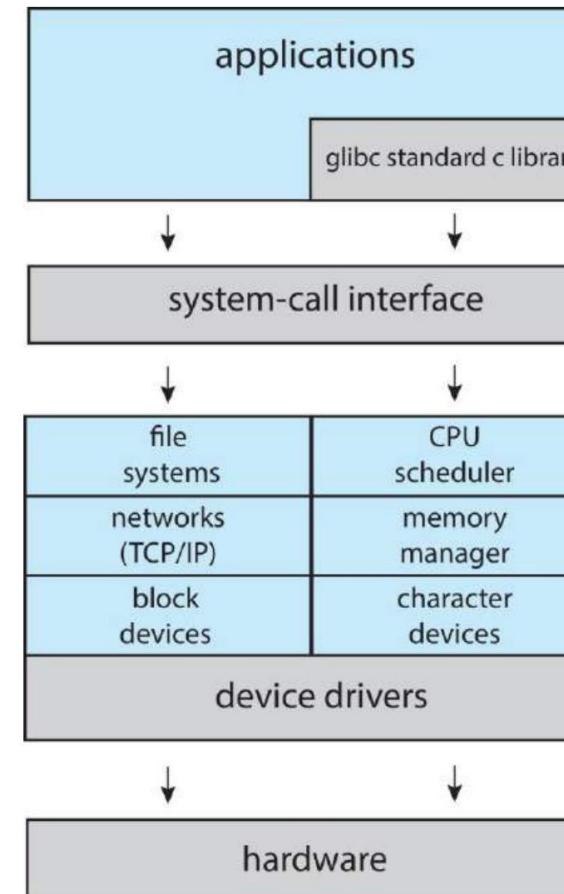
Más allá de lo simple pero no completamente en capas





Estructura del sistema Linux

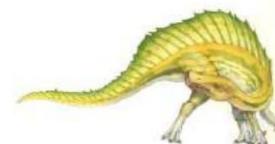
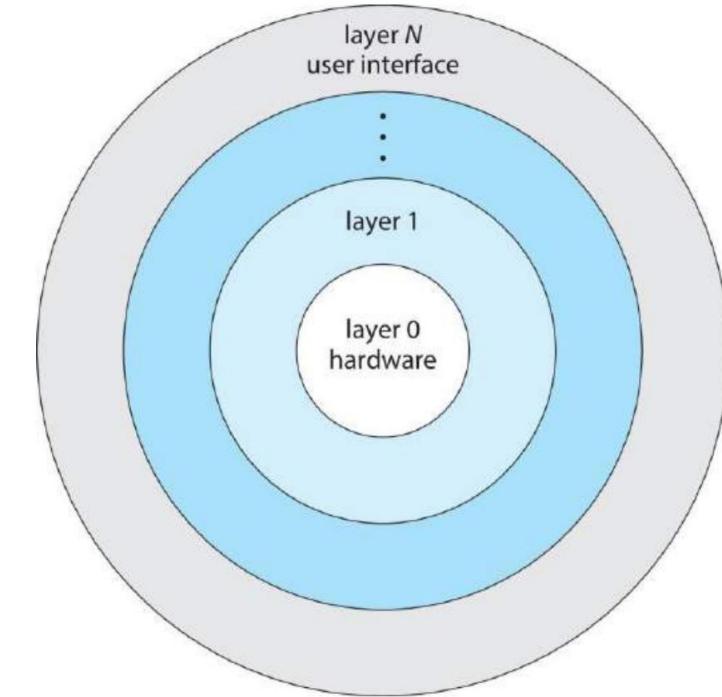
Diseño monolítico y modular





Enfoque en capas

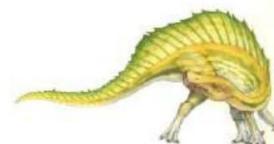
- El sistema operativo está dividido en varias capas (niveles), cada una construida sobre capas inferiores. La capa inferior (capa 0) es el hardware; la más alta (capa N) es la interfaz de usuario.
- Con la modularidad, las capas son seleccionadas de manera que cada uno utilice funciones (operaciones)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





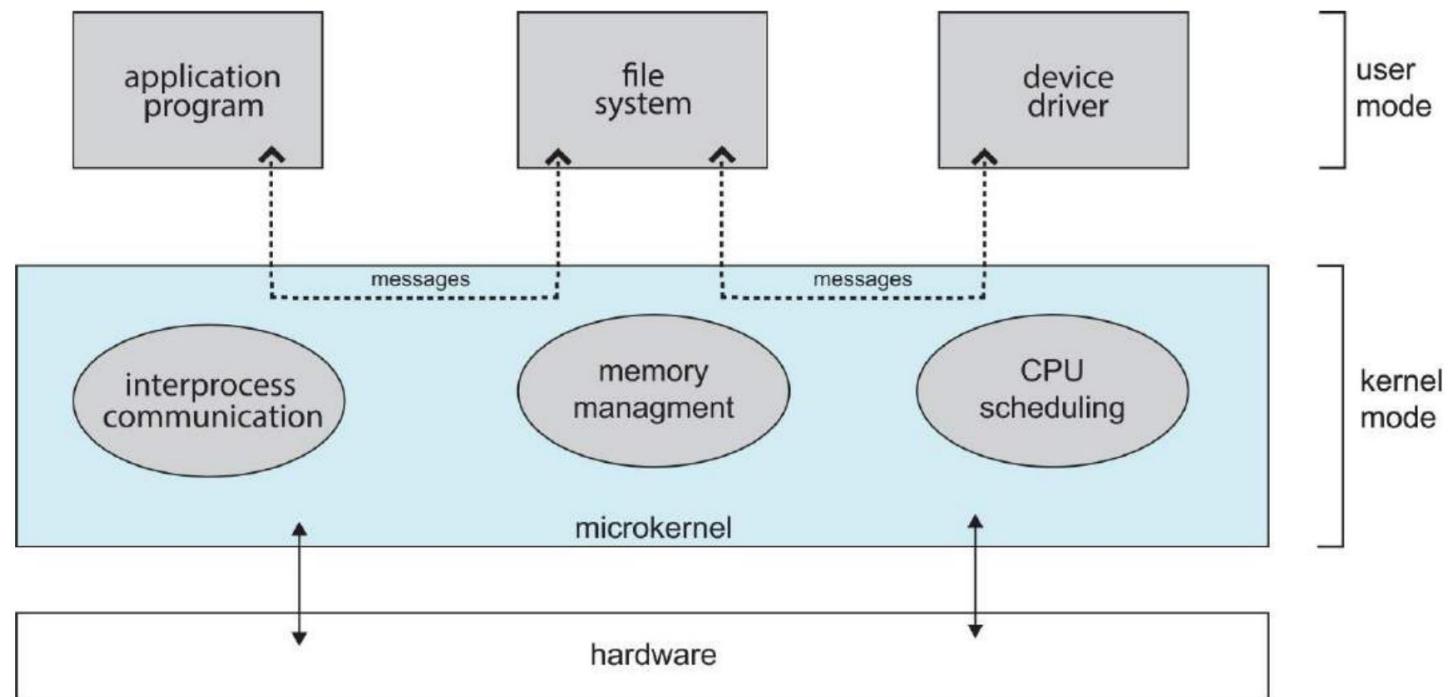
Micronúcleos

- Mueve todo lo posible desde el kernel al espacio del usuario.
- **Mach** es un ejemplo de **microkernel**
 - Kernel de Mac OS X ([Darwin](#)) basado parcialmente en Mach
- La comunicación se realiza entre módulos de usuario mediante [paso de mensajes](#)
- Beneficios:
 - Es más fácil extender un microkernel
 - Más fácil de portar el sistema operativo a nuevas arquitecturas
 - Más confiable (se ejecuta menos código en modo kernel)
 - Más seguro
- Detrimientos:
 - Sobrecarga de rendimiento de la comunicación entre el espacio del usuario y el espacio del kernel.





Estructura del sistema de micronúcleo





Módulos

- Muchos sistemas operativos modernos implementan **kernel cargable. módulos (LKM)**
 - Utiliza un enfoque orientado a objetos
 - Cada componente principal está separado
 - Cada uno habla con los demás a través de interfaces conocidas.
 - Cada uno se puede cargar según sea necesario dentro del kernel.
- En general, similar a las capas pero con más flexibilidad.
 - Linux, Solaris, etc.





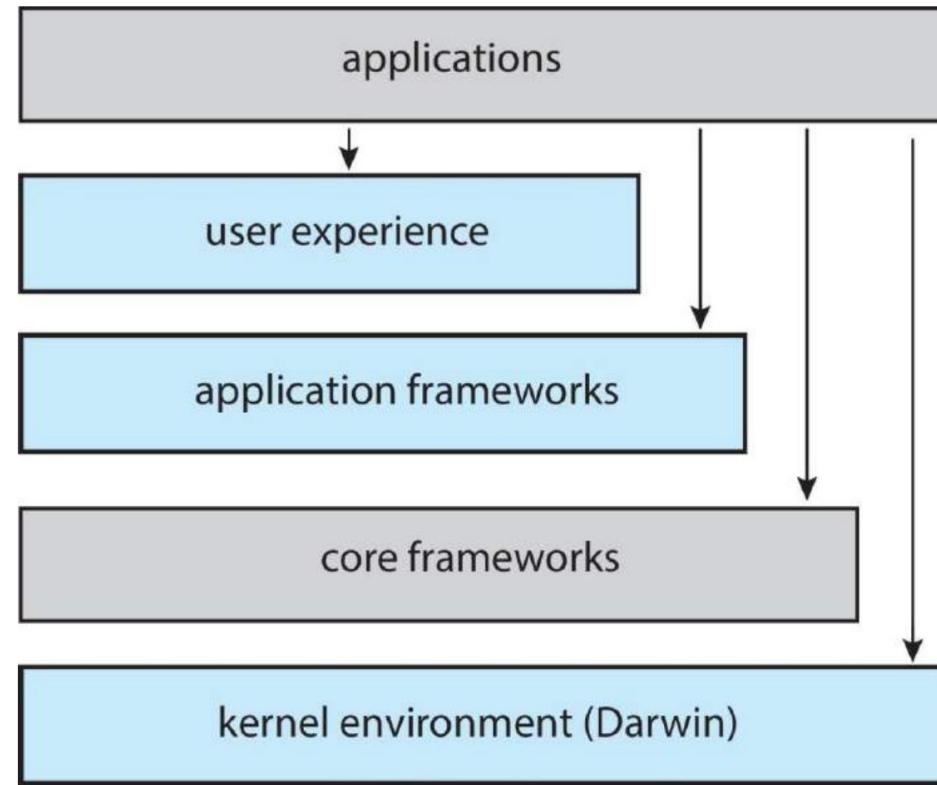
Sistemas híbridos

- La mayoría de los sistemas operativos modernos no son un modelo puro
 - Hybrid combina múltiples enfoques para abordar el rendimiento, necesidades de seguridad y usabilidad
 - Kernels de Linux y Solaris en el espacio de direcciones del kernel, por lo que son monolíticos y además modulares para la carga dinámica de funcionalidades.
 - Windows en su mayoría monolítico, además de microkernel para diferentes personalidades de subsistema
- Apple Mac OS X híbrido, en capas, [Aqua UI](#) más programación [Cocoa](#) ambiente
 - A continuación se muestra el kernel que consta del microkernel Mach y partes BSD Unix, además del kit de E/S y módulos cargables dinámicamente (llamados [extensiones del kernel](#)).



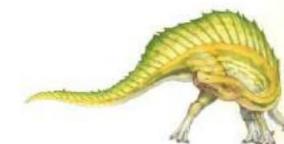
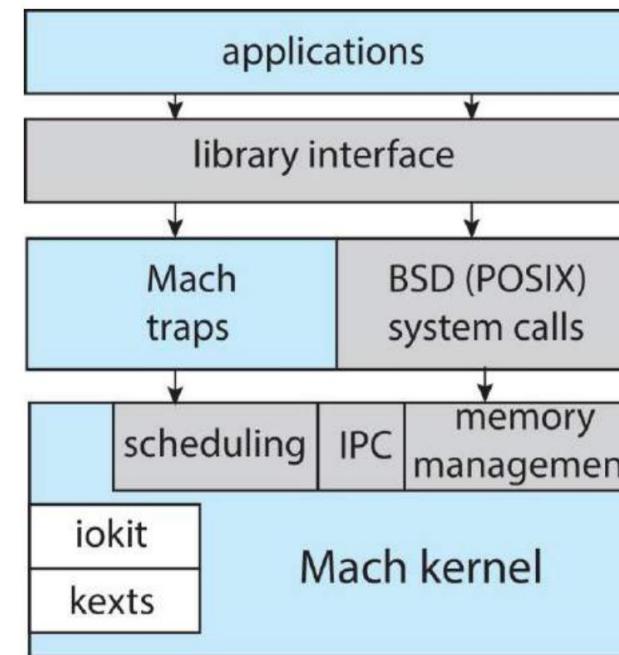


Estructura de macOS e iOS





Darwin





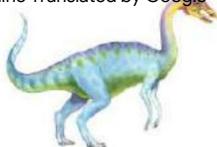
Androide

- Desarrollado por Open Handset Alliance (principalmente Google)
 - Código abierto
- Pila similar a IOS
- Basado en el kernel de Linux pero modificado
 - Proporciona gestión de procesos, memoria y controladores de dispositivos.
 - Agrega administración de energía
- El entorno de ejecución incluye un conjunto básico de bibliotecas y Dalvik máquina virtual
 - Aplicaciones desarrolladas en Java más API de Android

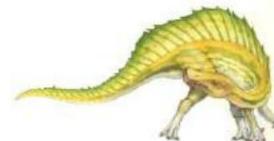
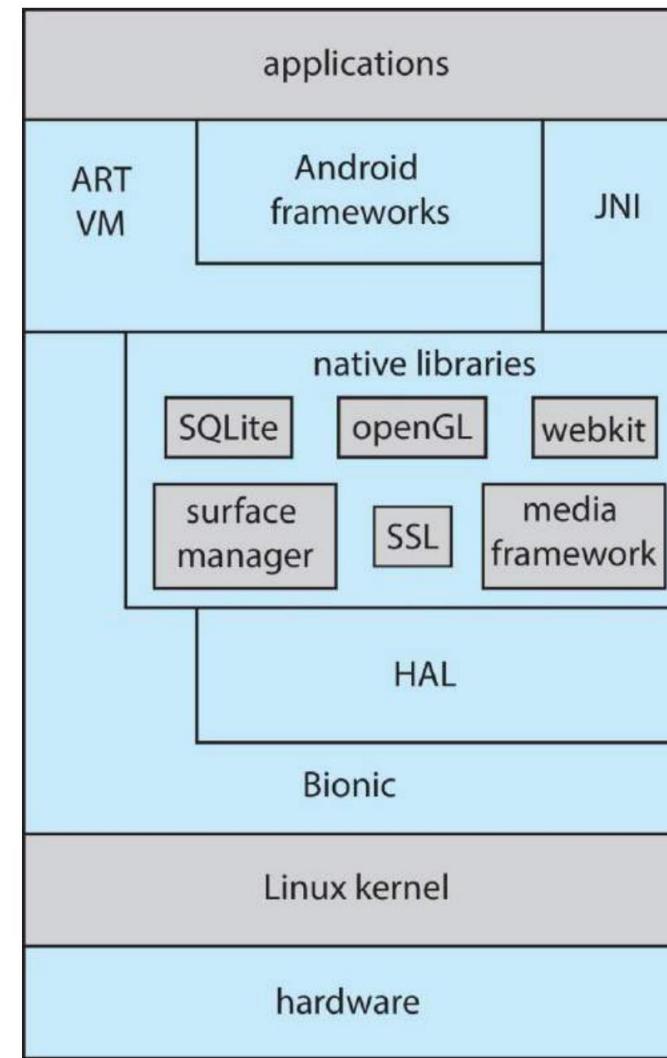
Archivos de clase Java compilados en código de bytes de Java y luego traducidos a ejecutables que se ejecutan en Dalvik VM

- Las bibliotecas incluyen marcos para navegador web (webkit), base de datos (SQLite), multimedia, libc más pequeña





Arquitectura de Android





Construyendo e iniciando un sistema operativo

- Sistemas operativos generalmente diseñados para ejecutarse en una clase de sistemas con variedad de periféricos
- Comúnmente, el sistema operativo ya instalado en el dispositivo comprado.
computadora
 - Pero puede construir e instalar algunos otros sistemas operativos.
 - **Si generas un sistema operativo desde cero**
 - Escribir el código fuente del sistema operativo
 - Configure el sistema operativo para el sistema en el que se ejecutará.
 - correr
 - Compilar el sistema operativo
 - Instalar el sistema operativo
 - Inicie la computadora y su nuevo sistema operativo.





Construyendo y arrancando Linux

- Descargar el código fuente de Linux (<http://www.kernel.org>)
- Configurar el kernel mediante “make menuconfig”
- Compile el kernel usando “make”
 - Produce vmlinuz, la imagen del núcleo.
 - Compile módulos del kernel a través de "hacer módulos"
 - Instale los módulos del kernel en vmlinuz mediante “make module_install”
 - Instale un nuevo kernel en el sistema mediante “make install”





Arranque del sistema

- Cuando se inicializa la energía en el sistema, la ejecución comienza en una memoria fija ubicación
- El sistema operativo debe estar disponible para el hardware para que éste pueda empezarlo
 - Pequeño fragmento de código: [cargador de arranque](#), [BIOS](#), almacenado en [ROM](#) o [EEPROM](#) localiza el kernel, lo carga en la memoria y lo inicia
 - A veces, un proceso de dos pasos donde [el bloque de arranque](#) en una ubicación fija se carga mediante código ROM, que carga el cargador de arranque desde el disco.
 - Los sistemas modernos reemplazan el BIOS con [Unified Extensible Interfaz de firmware \(UEFI\)](#)
- El cargador de arranque común, [GRUB](#), permite la selección del kernel desde múltiples discos, versiones, opciones de kernel
- El kernel se carga y el sistema se [ejecuta](#)
- Los cargadores de arranque frecuentemente permiten varios estados de arranque, como usuario único modo





Depuración del sistema operativo

- Depurar es encontrar y corregir errores o **fallos**.
- También **ajuste de rendimiento**
- El sistema operativo genera **archivos de registro** que contienen información de error.
- El fallo de una aplicación puede generar la captura de **archivos de volcado de núcleo**.
memoria del proceso
- Una falla del sistema operativo puede generar un archivo **de volcado de memoria** que contiene memoria del kernel
- Más allá de los fallos, el **ajuste del rendimiento** puede optimizar el rendimiento del sistema
 - A veces se utilizan listados de seguimiento de actividades, registrados para su análisis.
 - **La elaboración de perfiles** es un muestreo periódico de un puntero de instrucción para buscar tendencias estadísticas.

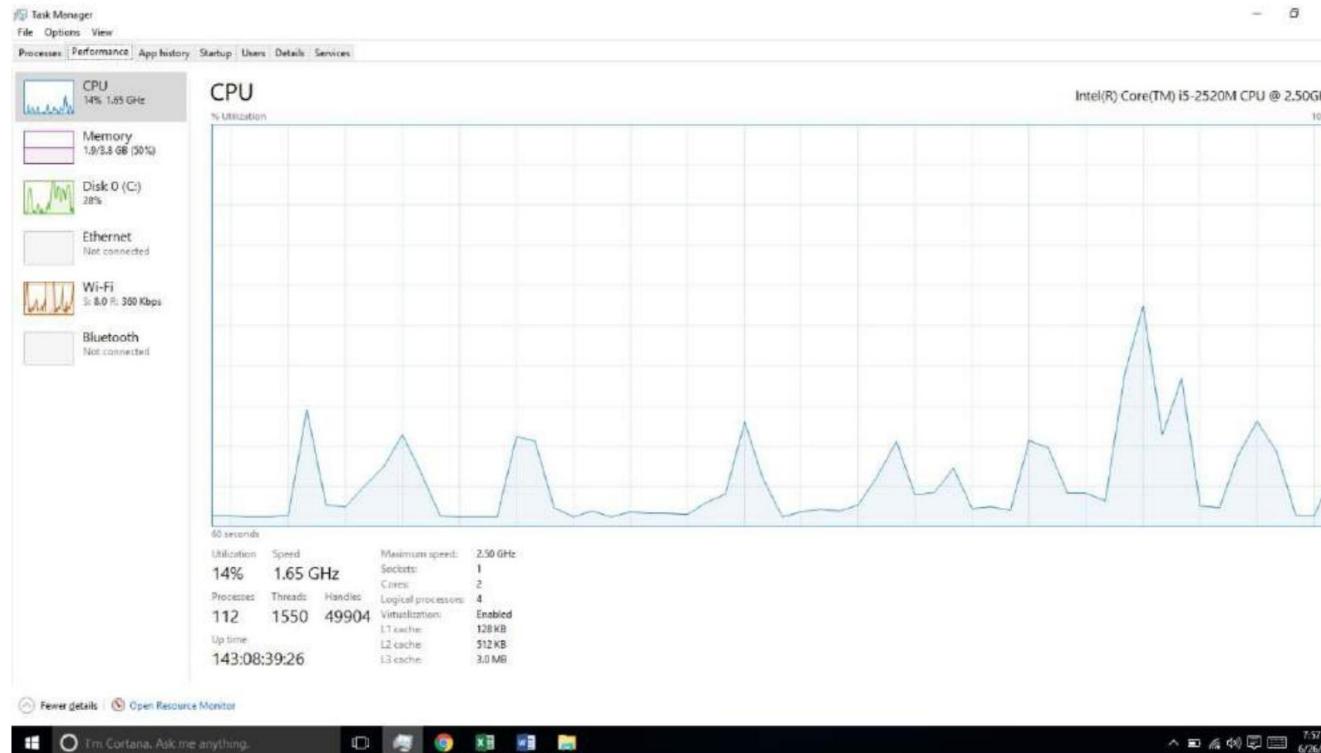
Ley de Kernighan: “La depuración es dos veces más difícil que escribir el código en primer lugar. Por lo tanto, si escribe el código de la forma más inteligente posible, por definición no será lo suficientemente inteligente como para depurarlo”.





La optimización del rendimiento

- Mejorar el rendimiento eliminando cuellos de botella
- El sistema operativo debe proporcionar medios para computar y mostrar medidas de comportamiento del sistema
- Por ejemplo, el programa “principal” o el Administrador de tareas de Windows





Rastreo

- Recopila datos para un evento específico, como los pasos involucrados en una invocación de llamada al sistema.
- Las herramientas incluyen
 - strace: rastrea llamadas al sistema invocadas por un proceso
 - gdb – depurador a nivel de fuente
 - perf – colección de herramientas de rendimiento de Linux
 - tcpdump – recopila paquetes de red



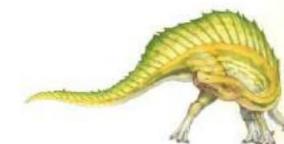


BCC

- Depurar las interacciones entre el nivel de usuario y el código del kernel es casi imposible sin un conjunto de herramientas que comprenda ambos y un instrumento para sus acciones.
- BCC (BPF Compiler Collection) es un completo conjunto de herramientas que proporciona funciones de seguimiento para Linux.
 - Ver también el DTrace original
- Por ejemplo, disksnoop.py rastrea la actividad de E/S del disco.

TIME(s)	T	BYTES	LAT(ms)
1946.29186700	R	8	0.27
1946.33965000	R	8	0.26
1948.34585000	W	8192	0.96
1950.43251000	R	4096	0.56
1951.74121000	R	4096	0.35

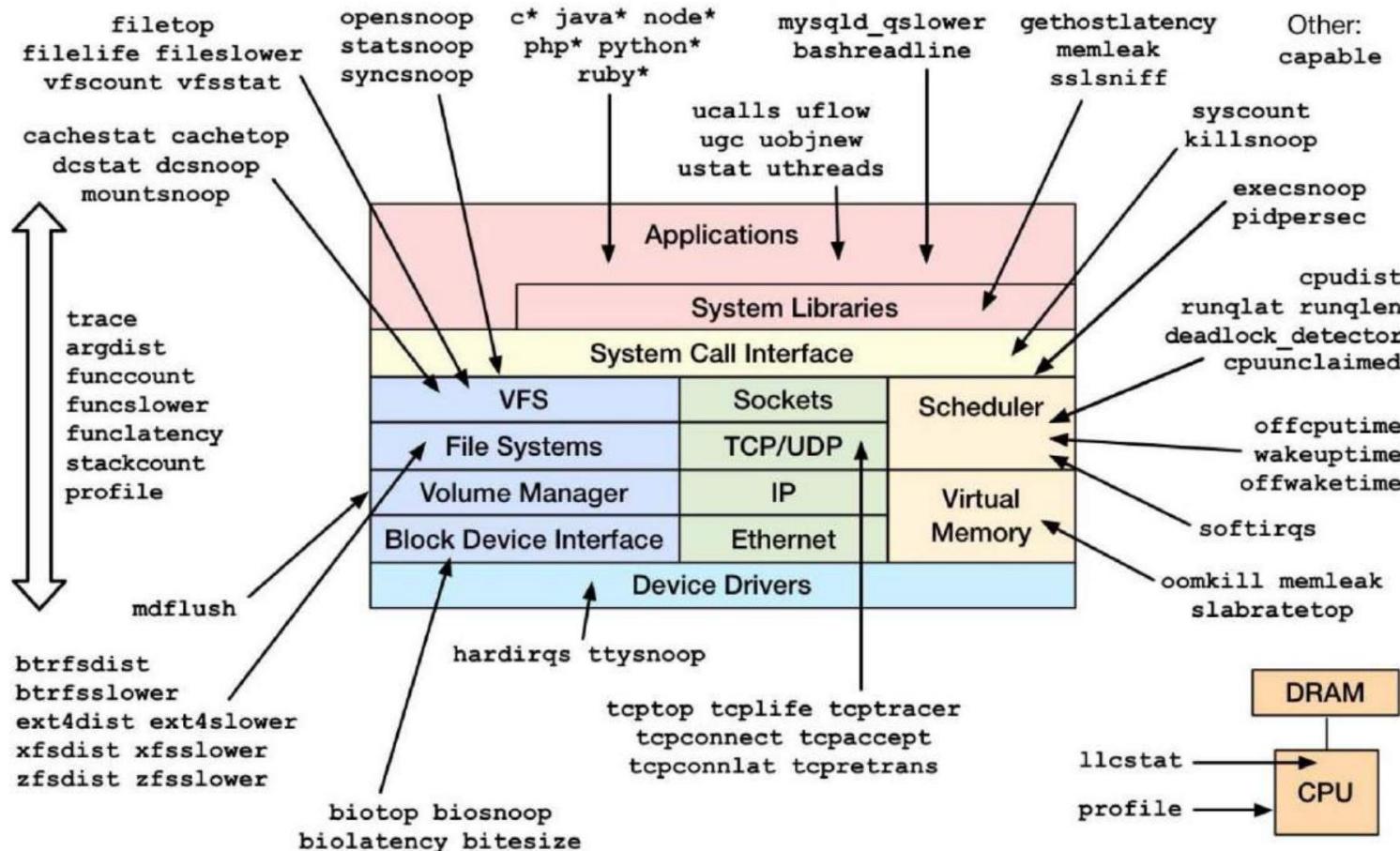
- Muchas otras herramientas (siguiente diapositiva)





Herramientas de seguimiento de Linux bcc/BPF

Linux bcc/BPF Tracing Tools



<https://github.com/iovisor/bcc#tools 2017>



Fin del Capítulo 2

