



Centro Universitario de Ciencias Exactas e Ingenieras.

Departamento para la integración ciber-humana.

Sistemas operativos.

Becerra Velázquez Violeta Rocío.

Yáñez Salazar Saul Emanuel.

220656484.

Ingeniería en computación.

D04.

Modelos de sistemas operativos.

Domingo 2 de feb. de 2025.

Modelos de sistemas operativos.

Monolítico.

En este modelo, es más tradicional y ampliamente utilizado en los sistemas operativos, en este enfoque todo el sistema operativo funciona como un solo programa en el modo núcleo, los servicios del sistema, como la gestión de procesos, la memoria y la entrada/salida, están integrados en único espacio de direcciones. Algunos ejemplos de sistemas operativos monolíticos incluyen Linux y Unix.

Cliente-servidor.

En este modelo, el sistema operativo se divide en procesos cliente y procesos servidor. Los clientes solicitan servicios específicos y los servidores los proporcionan. Este enfoque mejora la modularidad y la seguridad, ya que cada componente se ejecuta de manera independiente. En este caso, podríamos tomar como ejemplo el sistema operativo Minix.

Máquina virtual.

En este modelo, permiten ejecutar múltiples sistemas operativos en un solo hardware al abstraer los recursos físicos. Se basa en la creación de máquinas virtuales, cada una con su propio sistema operativo, lo que proporciona, así mismo y flexibilidad. Ejemplos que incluyen esto, son las máquinas ejecutadas en VMware y VirtualBox y el hipervisor de Microsoft Hyper-V.



Capas.

En este diseño, el sistema operativo se estructura en capas jerárquicas, cada una construida sobre la anterior. Cada capa tiene una función específica y solo interactúa con las capas adyacentes. Este modelo mejora la organización, y el mantenimiento del sistema operativo. Un ejemplo es el sistema operativo THE.



Historia de MacOS.

MacOS es el sistema operativo desarrollado por Apple Inc. para sus computadoras Mac. Su historia se remonta a los años 80, cuando Apple lanzó el Macintosh System Software, el precursor de macOS. Con el tiempo, el sistema evolucionó significativamente a través de diversas versiones y mejoras.

Los inicios: System 1 a Mac OS 9

En 1984, Apple presentó el Macintosh con System 1, un sistema operativo basado en una interfaz gráfica de usuario que revolucionó la computación personal. Durante los años siguientes, el sistema evolucionó hasta Mac OS 9, la última versión de la línea clásica de Mac OS, lanzada en 1999. Sin embargo, este sistema carecía de muchas características avanzadas como memoria protegida y multitarea apropiativa.

El salto a Mac OS X

En 2001, Apple lanzó Mac OS X, una reescritura completa basada en el sistema operativo NeXTSTEP, que Apple adquirió tras comprar NeXT, la empresa fundada por Steve Jobs. Mac OS X introdujo un núcleo basado en UNIX (XNU) y una interfaz de usuario moderna llamada Aqua. Su arquitectura híbrida permitió mayor estabilidad, seguridad y rendimiento en comparación con las versiones anteriores.

Transiciones de hardware

Apple ha realizado múltiples transiciones de hardware a lo largo de la historia de macOS:

- En 2006, la compañía migró de procesadores PowerPC a Intel, permitiendo a los usuarios ejecutar Windows en Mac mediante Boot Camp.
- En 2020, Apple anunció su transición a los chips Apple Silicon basados en arquitectura ARM, comenzando con el M1, ofreciendo mejoras en eficiencia y rendimiento.



El primer Hackintosh

Uno de los eventos más significativos en la comunidad de entusiastas fue la creación del primer "Hackintosh", una computadora no fabricada por Apple ejecutando macOS. En 2005, antes de la transición oficial a Intel, desarrolladores lograron ejecutar versiones modificadas de Mac OS X en hardware no autorizado, abriendo la posibilidad de instalar macOS en PC estándar. A partir de la transición a Intel, la práctica de crear Hackintosh se volvió más accesible, dando lugar a una comunidad activa que sigue buscando maneras de ejecutar macOS en hardware no oficial.

macOS en la actualidad

Desde 2016, Apple renombró Mac OS X a macOS, alineándolo con sus otros sistemas operativos como iOS, iPadOS y watchOS. Actualmente, macOS sigue evolucionando con mejoras en rendimiento, integración con dispositivos Apple y nuevas funciones enfocadas en la privacidad y seguridad del usuario.

macOS ha recorrido un largo camino desde sus inicios, consolidándose como uno de los sistemas operativos más influyentes en la historia de la computación.

Servicios que ofrece MacOS.

MacOS al ser un sistema basado en Unix, ofrece lo que ofrece cualquier sistema operativo para computadoras, seguridad, administración de ram, archivos, procesos y comunicación entre otros dispositivos externos, pero además de esto, ofrecen otras mejoras que lo destacan sobre los otros servicios como lo es.

Almacenamiento en la nube por iCloud.

Asistente de voz personal con funciones de control de voz.

Desarrollo de aplicaciones multiplataforma para diferentes sistemas operativos, como lo es para Android, iOS y todo el ecosistema que tiene Apple.

Objetivos de macOS

macOS tiene varios objetivos clave que guían su desarrollo y evolución:

Seguridad y privacidad: Garantizar la protección de los datos del usuario mediante funciones avanzadas como cifrado, protección contra malware y controles de privacidad estrictos.

Experiencia de usuario intuitiva: Ofrecer una interfaz gráfica elegante, fácil de usar y consistente en todos los dispositivos Apple.

Integración con el ecosistema Apple: Facilitar la conectividad fluida entre dispositivos Apple, permitiendo la continuidad y sincronización sin interrupciones.

Alto rendimiento y estabilidad: Optimizar el uso de hardware para ofrecer velocidad, eficiencia energética y fiabilidad.

Compatibilidad con aplicaciones profesionales y de consumo: Soportar herramientas avanzadas para diseñadores, programadores, editores de video y otros profesionales, además de proporcionar una amplia variedad de aplicaciones para el usuario general.

Innovación constante: Incorporar nuevas tecnologías y mejoras que mantengan macOS a la vanguardia de la industria informática.

Funciones de macOS

macOS incluye diversas funciones que mejoran la experiencia del usuario y la eficiencia del sistema.

Finder: Explorador de archivos con una interfaz intuitiva y funciones avanzadas de organización.

Mission Control: Vista general de todas las ventanas abiertas, escritorios virtuales y aplicaciones en ejecución.

Launchpad: Acceso rápido y visual a todas las aplicaciones instaladas en el sistema.

Split View: Permite dividir la pantalla entre dos aplicaciones para mejorar la productividad.

Automator y Atajos: Herramientas de automatización para simplificar tareas repetitivas.

AirDrop: Función para compartir archivos rápidamente entre dispositivos Apple.

Sidecar: Extiende la pantalla de Mac a un iPad para usarlo como segunda pantalla.

Modo oscuro: Interfaz con tonos oscuros para reducir el cansancio visual y mejorar la estética del sistema.

Compatibilidad con aplicaciones de iOS: Permite ejecutar apps diseñadas para iPhone y iPad en Mac con procesadores Apple Silicon.

Con estas funciones, macOS ofrece una experiencia optimizada para la productividad, creatividad y conectividad en el ecosistema Apple.

Estructura de macOS

macOS está compuesto por varias capas que trabajan en conjunto para ofrecer un sistema estable, seguro y eficiente:

Núcleo (Kernel XNU):

Es la base del sistema, combinando componentes de UNIX y tecnologías propias de Apple. Maneja procesos, memoria y dispositivos de hardware.

Capa de Servicios del Sistema:

Incluye frameworks esenciales como Core Foundation y Core Services. Gestiona funciones de red, bases de datos, multimedia y más.

Capa de Gráficos y Multimedia:

Compuesta por tecnologías como Metal, Core Animation y Core Image. Responsable del procesamiento gráfico y la renderización de interfaz.

Capa de Frameworks de Aplicaciones:

Proporciona herramientas para el desarrollo de software, incluyendo Cocoa y SwiftUI. Permite la creación de aplicaciones con interfaces modernas y eficientes.

Interfaz de Usuario (Aqua):

La capa visible para los usuarios, con elementos gráficos elegantes y animaciones fluidas. Incluye Finder, Dock, Mission Control y otros componentes clave.

¿Qué significa JCL?

JCL significa Job Control Language. Es un lenguaje de scripting utilizado en sistemas mainframe de IBM, específicamente en z/OS para controlar la ejecución de trabajos por lotes.

Escribe la diferencia entre el procesamiento por lotes y el de procesamiento por multiprogramación.

La diferencia entre procesamiento por lotes y procesamiento por lotes con multiprogramación radica en la manera en que se gestionan los trabajos y los recursos del sistema para maximizar la eficiencia.

Mientras que el procesamiento de lotes, los trabajos se ejecutan uno tras otro, no hay aprovechamiento activo de los recursos del sistema, mientras un trabajo está esperando, no se aprovecha toda la capacidad del procesamiento del sistema, mientras que con multiprogramación varios trabajos residen en la memoria al mismo tiempo y el procesador cambia de uno a otro de forma rápida para aprovechar los periodos de inactividad, el procesador no se detiene cuando un trabajo está esperando, si un trabajo está en espera, otro trabajo puede aprovechar el tiempo de espera y utilizar el procesador, lo cual significa mejor utilización de los recursos del sistema, ya que se aprovecha la capacidad de procesamiento en todo momento.

Escribe una de las utilidades de la interrupción int86 en C.

Una de las utilidades más comunes de la interrupción int86 en C es la manipulación de información del sistema en entornos DOS, como la obtención de datos sobre el estado del hardware o la interacción con periféricos. A través de int86, es posible invocar servicios proporcionados por el sistema operativo o la BIOS.

```
app.c x

#include <stdio.h>
#include <dos.h> // Necesario para usar int86 y trabajar con interrupciones

void obtenerVersionDOS() {
    union REGS r;

    r.h.ah = 0x30; // Función 0x30: Obtener versión de DOS

    // Llamada a la interrupción 0x21 para obtener la versión de DOS
    int86(0x21, &r, &r);

    // La versión de DOS se encuentra en los registros:
    // r.h.al contiene la versión mayor (1, 2, 3, etc.)
    // r.h.ah contiene la versión menor (por ejemplo, 0, 1, 2)
    printf("Versión de DOS: %d.%d\n", r.h.al, r.h.ah);
}
```

¿Qué hace al función kbhit en C?

La función **kbhit** en C es parte de la biblioteca **conio.h** (consola de entrada/salida) y se utiliza para detectar si una tecla ha sido presionada en el teclado sin que el programa se detenga a esperar por esa entrada. Es una función muy útil cuando se desea realizar un procesamiento en tiempo real o interactuar con el usuario, pero sin que el flujo del programa se vea interrumpido esperando por una entrada del teclado.

```
app.c

#include <conio.h>
#include <stdio.h>

int main() {
    printf("Presiona cualquier tecla para salir...\n");

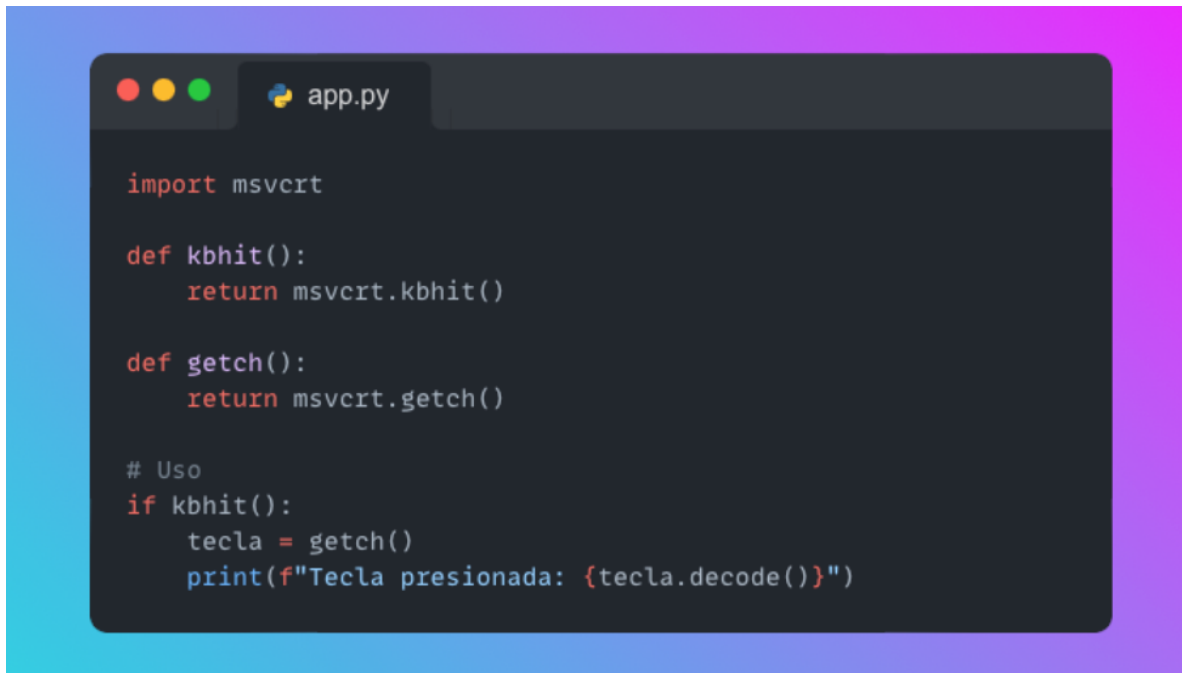
    // Bucle que espera a que se presione una tecla
    while (!kbhit()) {
        // Aquí podrían ir otras tareas que quieras ejecutar mientras esperas
    }

    // Una vez que se presiona una tecla, se captura y termina el programa
    char tecla = getch(); // Obtener la tecla presionada
    printf("Tecla presionada: %c\n", tecla);

    return 0;
}
```

Sistemas operativos.

Equivalencias en otros lenguajes de programación.

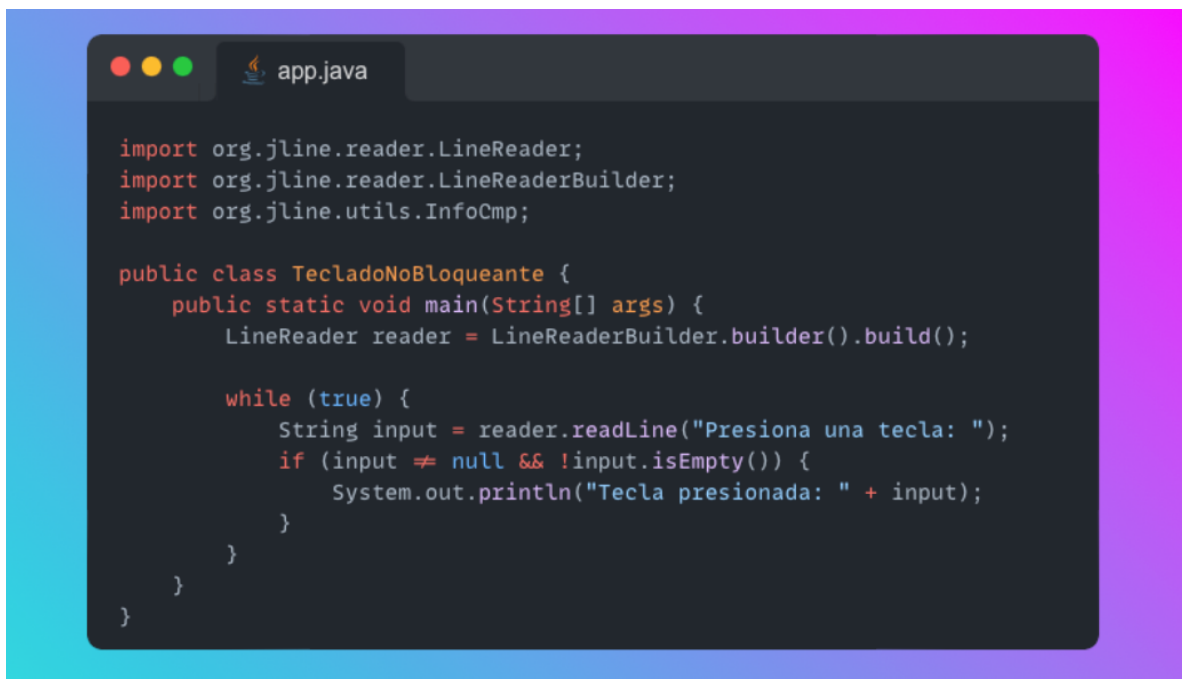


```
import msvcrt

def kbhit():
    return msvcrt.kbhit()

def getch():
    return msvcrt.getch()

# Uso
if kbhit():
    tecla = getch()
    print(f"Tecla presionada: {tecla.decode()}")
```



```
import org.jline.reader.LineReader;
import org.jline.reader.LineReaderBuilder;
import org.jline.utils.InfoCmp;

public class TecladoNoBloqueante {
    public static void main(String[] args) {
        LineReader reader = LineReaderBuilder.builder().build();

        while (true) {
            String input = reader.readLine("Presiona una tecla: ");
            if (input != null && !input.isEmpty()) {
                System.out.println("Tecla presionada: " + input);
            }
        }
    }
}
```