SISTEMAS OPERATIVOS

ACTIVIDAD DE APRENDIZAJE 3
TAREA 2

Profesor:
VIOLETA DEL ROCIO BECERRA VELAZQUEZ

VERDUZCO ROSALES LUIS ENRIQUE
223992388
Ingenieria en Computación

07/09/25

CUCEI DIVTIC D04

ÍNDICE

TABLA IMAGENES

Modelos de Sistemas Operativos

1. Monolítico:
   En un sistema operativo monolítico, el kernel es el núcleo del sistema operativo y es responsable de la gestión de los recursos de hardware y software del sistema. El kernel se ejecuta en modo kernel, lo que significa que tiene acceso completo a los recursos del sistema y puede ejecutar cualquier instrucción del procesador.

   El kernel es un único programa grande donde todos los componentes (gestión de memoria, sistema de archivos, controladores, etc.) se ejecutan en el espacio del kernel. La comunicación entre componentes es rápida, pero un error en cualquier parte puede hacer caer todo el sistema.

   - Ejemplo: Xv6. Es un teaching OS moderno, clon didáctico del Unix V6, usado en universidades para enseñar cómo funciona un SO monolítico clásico.

2. Cliente-Servidor (Microkernel):
   El kernel (microkernel) es mínimo y solo se encarga de la comunicación entre procesos (IPC) y la gestión esencial de hardware. Los demás servicios (servidores) se ejecutan como procesos en espacio de usuario, lo que ofrece mayor estabilidad y modularidad.

   Esta estrategia de delegar otros servicios a procesos que se realizan en el equipo del usuario, contrasta con los sistemas monolíticos, en los cuales el kernel maneja múltiples servicios.

   Las ventajas clave de los microkernels incluyen mayor seguridad y fiabilidad debido a la reducción de código con privilegios elevados, y la modularidad, que facilita el mantenimiento y actualización de componentes. Sin embargo, presentan desventajas como posible impacto en el rendimiento y mayor complejidad en el diseño y desarrollo.

   - Ejemplo: Minix 3. Diseñado específicamente para ser extremadamente fiable. Si un servidor de drivers (ej. el de la tarjeta de red) falla, el microkernel puede detectarlo y reiniciarlo automáticamente sin afectar al resto del sistema.

3. Máquina Virtual:
   El SO principal (host) crea una capa de abstracción (hypervisor) que emula hardware, permitiendo ejecutar múltiples sistemas operativos invitados (guests) de forma aislada y simultánea.

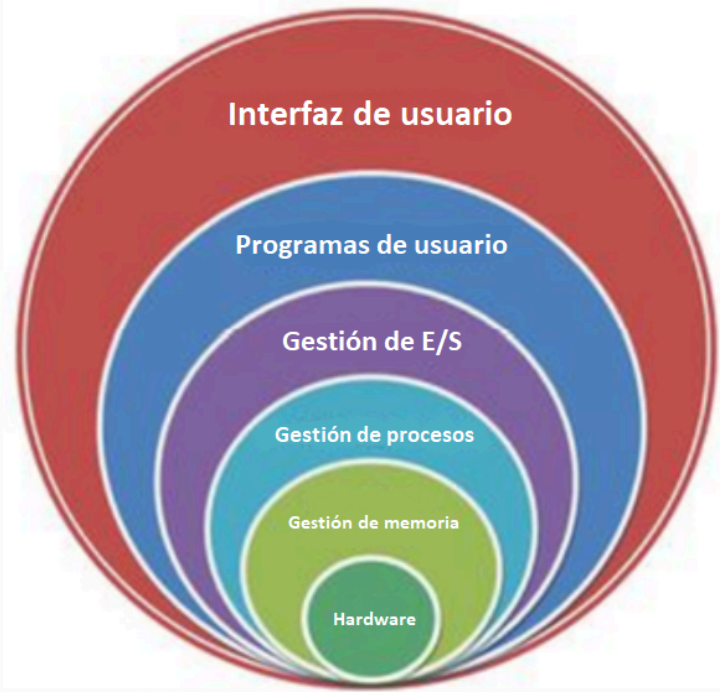
- Ejemplo: Xen. Un hipervisor de código abierto, muy popular en entornos de computación en la nube y servidores por su eficiencia y capacidad de paravirtualización.

4. Capas (En capas):
   El sistema se organiza en niveles o anillos de abstracción concéntricos. Cada capa solo puede usar servicios de las capas inferiores, proporcionando un diseño ordenado y fácil de depurar.

   La capa más interna o inferior capa 0 corresponde al hardware, muestran que la más alta o externa corresponde a la interfaz del usuario.

   El primer sistema construido de esta manera fue el sistema THE (Technische Hogeschool Eindhoven), desarrollado en Holanda por E. W. Dijkstra (1968) y sus estudiantes; el cual consta de 6 capas que son:

   Sistema operativo por capas



- Ejemplo: THE Operating System. Desarrollado por Edsger Dijkstra y su equipo en los años 60. Fue pionero en el concepto de capas y en la verificación formal de software, aunque nunca se usó comercialmente.

5. Híbrido: Combina las ideas de los modelos monolítico y de microkernel para lograr un balance entre rendimiento (ejecutando algunos servicios clave en espacio del kernel) y estabilidad (manteniendo otros servicios en espacio de usuario).

    - Ejemplo: BeOS. Diseñado para multimedia y alto rendimiento en hardware de la época. Su kernel modular (kernel de "objetos") era híbrido, permitiendo un gran paralelismo y capacidad de respuesta en tiempo real.

Los 8 Sistemas Operativos:

1. Haiku OS (2001 - primera versión alpha):
   Inspirado en BeOS. Su objetivo es ser rápido, eficiente, fácil de usar y responsive, especialmente para computación personal y multimedia.

    - Característica principal: Sistema de archivos journaling (BFS) con base de datos integrada para metadatos.

2. ReactOS (1998 - primera versión pública):
   Un proyecto de código abierto que busca ser un clon binariamente compatible con Windows NT y sus aplicaciones/controladores.

    - Característica principal: Compatibilidad con APIs de Windows (Win32) sin usar código de Microsoft.

3. Plan 9 from Bell Labs (1992 - primera versión pública):
   Creado por los mismos laboratorios Bell que crearon Unix. Lo consideran el "verdadero sucesor de Unix".

    - Característica principal: Todo es un sistema de archivos (interfaces de red, procesos, ventanas), facilitando la distribución y transparencia de recursos.

4. TempleOS (2013 - como "J Operating System"):
   Creado por una sola persona, Terry A. Davis. Es un sistema operativo de dominio público para x86-64.

    - Característica principal: Diseñado para ser un "templo" simple y seguro, con gráficos de 640x480 en 16 colores y un compilador integrado.

5. Inferno (1996 - por Bell Labs):
   Derivado de Plan 9, pero diseñado para ser extremadamente portable y ejecutarse como un sistema independiente o como una aplicación dentro de otro SO.

   - Característica principal: Su máquina virtual, Dis, permite que las aplicaciones (escritas en Limbo) se ejecuten igual en cualquier hardware donde corra Inferno.

6. SerenityOS (2018 - primera versión):
   Un sistema operativo gráfico moderno desde cero, creado principalmente por Andreas Kling.

   - Característica principal: Desarrollado con amor y detalle, incluye su propio kernel, navegador web (Ladybird) y entorno de escritorio, todo con una estética vintage.

7. MorphOS (2000 - primera versión):
   Un sistema operativo de alto rendimiento y eficiencia energética, compatible con PowerPC y algunas placas base x86.

   - Característica principal: Compatibilidad binaria con aplicaciones de AmigaOS a través de su capa de emulación "Trinity", manteniendo la filosofía de Amiga.

8. KolibriOS (2004 - fork de MenuetOS):
   Uno de los SO más pequeños y rápidos del mundo, escrito casi enteramente en ensamblador (FASM).

   - Característica principal: Se ejecuta desde un disquete o USB y arranca en segundos, incluyendo una interfaz gráfica completa con aplicaciones como navegador web y reproductor multimedia.

Sistema Operativo a Elección:

Plan 9 de Bell Labs

1. Historia:

   Desarrollado a finales de los 80s y lanzado comercialmente en 1992 por los Laboratorios Bell (creadores de Unix). Nació de la insatisfacción con las limitaciones de Unix en entornos distribuidos y de red. Aunque no tuvo éxito comercial, su influencia en conceptos de sistemas distribuidos es profunda.

2. Servicios que presta:

    - 9P Protocol: Un protocolo de comunicación para acceder a recursos remotos como si fueran locales.

    - Sistema de archivos unificado: No solo archivos, sino también dispositivos, interfaces de red y la propia interfaz de usuario se representan como sistemas de archivos.

    - Espacios de nombres por proceso: Cada proceso puede tener una vista personalizada y aislada del sistema de archivos global.

    - Computación distribuida transparente: Permite ensamblar un entorno de trabajo personal usando recursos (CPU, almacenamiento) de varias máquinas en la red de forma seamless.

3. Objetivos:

    - Corregir los defectos de diseño de Unix, especialmente en el manejo de recursos distribuidos.

    - Proporcionar un modelo uniforme para representar y acceder a todos los recursos, locales o remotos.

    - Facilitar la construcción de sistemas distribuidos y entornos de computación colaborativa.

4. Funciones:

    - Gestionar recursos locales y distribuidos a través del protocolo 9P.
    - Proporcionar un entorno de programación concurrente y distribuido.
    - Ofrecer una interfaz de usuario gráfica (rio) o de texto a través de su sistema de archivos.

5. Estructura:
   Plan 9 no sigue estrictamente un modelo monolítico o de microkernel. Es más bien un sistema distribuido kernel-centric. El kernel gestiona los recursos locales, pero la clave es el protocolo 9P que permite conectar múltiples kernels y servidores de recursos (file servers, CPU servers, auth servers) para formar un solo sistema virtual distribuido.

Preguntas

1. ¿Qué significa JCL?

    - Job Control Language (Lenguaje de Control de Trabajos). Era un lenguaje de scripting usado principalmente en mainframes IBM OS/360 y sucesores para especificar los recursos, pasos y datos necesarios para ejecutar un programa ("job") por lotes.

2. Escriba la diferencia entre el procesamiento por lotes y el de procesamiento por lotes con multiprogramación.

    - Procesamiento por lotes simple (Batch):
      Una serie de trabajos se ejecutan sin intervención humana en el sistema de procesamiento por lotes. En este conjunto, trabajos con necesidades similares se agrupan y se introducen en la computadora para su ejecución.

    - Procesamiento por lotes con multiprogramación:
      Un sistema operativo multiprogramación permite ejecutar múltiples procesos mediante la monitorización de sus estados y la alternancia entre ellos. Ejecuta múltiples programas para evitar la infrautilización de la CPU y la memoria.

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

    - int86() era una función de la biblioteca <dos.h> en compiladores como Turbo C. Su utilidad era invocar servicios de la BIOS (Basic Input/Output System).

    - Por ejemplo, se usaba para cambiar el modo de video de la tarjeta gráfica (int 0x10) o para leer directamente del teclado a un nivel muy bajo (int 0x16), evitando el sistema operativo.

4. ¿Para qué sirve la función Kbhit?

    - kbhit() (keyboard hit) es una función (típica de <conio.h>) que verifica de manera no bloqueante si una tecla ha sido presionada en el buffer del teclado. No espera a que el usuario presione una tecla; devuelve un valor distinto de cero (true) si hay una tecla esperando y cero (false) si no la hay. Es fundamental para videojuegos o aplicaciones en tiempo real que no pueden detenerse a esperar una entrada.

5. Investigue el equivalente de Kbhit (utilizado en c) en otros dos lenguajes de programación y escríbalos.

- Python (usando la biblioteca msvcrt solo en Windows):

```
import msvcrt
if msvcrt.kbhit():
key = msvcrt.getch()
print(f"Tecla presionada: {key}")
```

- C# (.NET):

```
if (Console.KeyAvailable) // <-- Este es el equivalente a kbhit()
{
    ConsoleKeyInfo key = Console.ReadKey(true);
    Console.WriteLine($"Tecla presionada: {key.KeyChar}");
}
```

Bibliografía

González, H. (s.f.). ¿Qué es un sistema operativo monolítico? Herschel González. Recuperado de: https://herschelgonzalez.com/que-es-un-sistema-operativo-monolitico/

ITDO. (2021, 19 julio). Arquitectura de Microkernel: Ventajas y desventajas en el desarrollo de sistemas operativos. Recuperado de: https://www.itdo.com/blog/arquitectura-de-microkernel-ventajas-y-desventajas-en-el-desarrollo-de-sistemas-operativos/

Rodríguez, I. (s.f.) 1.2.1 Tipos: Monolítico, por capas, micro-núcleo (microkernels), máquina virtual y exokernels. Comunidad ESCOM IPN. Recuperado de: https://www.comunidad.escom.ipn.mx/israelsr/RDD_SO_U1/pages/1-2-1-tipos-monol%C3%ADtico-por-capas-micro-n%C3%BAcleo-microkernels-m%C3%A1quina-virtual-y-exokernels.html

Villalba, E. (2022). BI-OS Sistema Operativo. Studocu. Recuperado de: https://www.studocu.com/latam/document/universidad-americana-paraguay/introduccion-al-analisis-de-sistema/bi-os-sistema-operativo/92689295

Muñoz, J. (2025, 24 marzo). ReactOS se actualiza: el 'Windows' de código abierto sigue vivo. MuyComputer. Recuperado de:

https://www.muycomputer.com/2025/03/24/reactos-se-actualiza-el-windows-de-codigo-abierto-sigue-vivo/

Casterán, H. (s.f.). Plan 9 from Bell Labs. Facultad de Ciencias Exactas, Ingeniería y Agrimensura - UNR. Recuperado de: https://www.fceia.unr.edu.ar/~hcaste/plan9.html

García, J. (2021, 17 agosto). TempleOS, la rareza de un sistema operativo denotado como revolucionario (I). Genbeta. Recuperado de: https://www.genbeta.com/sistemas-operativos/templeos-rareza-sistema-operativo-denostado-como-revolucionario-1

Juancho. (2023, 3 noviembre). TempleOS: El sistema operativo inspirado por Dios que desafió las expectativas. Tecnología con Juancho. Recuperado de: https://tecnologiaconjuancho.com/templeos-el-sistema-operativo-inspirado-por-dios-que-desafio-las-expectativas/

Kiddle. (s.f.). Inferno (sistema operativo). Kiddle. Recuperado de: https://ninos.kiddle.co/Inferno_(sistema_operativo)

Muñoz, J. (2021, 18 agosto). SerenityOS: un Unix alternativo. MuyComputer. Recuperado de: https://www.muycomputer.com/2021/08/18/serenityos-unix-alternativo/

The MorphOS Development Team. (s.f.). Introduction. MorphOS Team. Recuperado de: https://www.morphos-team.net/intro

LinuxMind. (2024, 15 mayo). Guía completa del SO KolibriOS: cómo funciona, orientación y curiosidades. Recuperado de: https://linuxmind.dev/es/2024/05/15/guia-completa-del-so-kolibrios-como-funciona-orientacion-y-curiosidades/

IBM. (2021, 27 octubre). What is JCL? Recuperado de: https://www.ibm.com/docs/en/zos-basic-skills?topic=sdsf-what-is-jcl

GeeksforGeeks. (2024, 28 junio). Difference between Batch Processing OS and Multiprogramming OS. Recuperado de: https://www.geeksforgeeks.org/operating-systems/difference-between-batch-processing-os-and-multiprogramming-os/

Bobo, B. (2008, 3 noviembre). Counterpart of kbhit() in C. Bobobobo's Weblog. Recuperado de: https://bobobobo.wordpress.com/2008/11/03/counterpart-of-kbhit-in-c/

Lum, B., & Liccini, M. (2014). Plan 9: The Way the Future Was. 6.824: Computer Systems Research. https://css.csail.mit.edu/6.824/2014/papers/plan9.pdf

Pike, R., Presotto, D., Dorward, S., Flandrena, B., Thompson, K., Trickey, H., & Winterbottom, P. (1995). Plan 9 from Bell Labs. Computing Systems. https://www.usenix.org/legacy/publications/compsystems/1995/sum_pike.pdf

Lundh, F. (2001). The readline module. In Python Standard Library. O'Reilly Media. Recuperado de: https://www.oreilly.com/library/view/python-standard-library/0596000960/ch12s11.html