Centro Universitario de Ciencias Exactas e Ingenierías.

Departamento de Ciencias Computacionales.

Materia:

Sistemas Operativos.

Profesora:

Becerra Velázquez Violeta del Rocío.

Alumno:

Maldonado Melendez Diego Alberto.

Código:

221977845.

Carrera:

Ingeniería en Computación.

Sección:

D04

Título de la investigación:

Modelos de Sistemas Operativos.

Fecha:

27 de agosto de 2023.

Índice

Tabla de Imágenes. 3
1. Modelo de Sistemas Operativos. 4
1.1 Sistema monolítico. 4
1.2 Modelo Cliente-Servidor. 5
1.3 Máquina virtual. 6
1.4 Capas. 7
1.5 Híbrido. 7
2. Sistema Operativo macOS. 8
2.1 Servicios que presta. 9
2.2 Objetivos. 9
2.3 Funciones. 10
2.4 Estructura. 10
3. Preguntas. 11
3.1 ¿Qué significa JCL? 11
3.2 Diferencia entre procesamiento por lotes y procesamiento por lotes con multiprogramación. 11
3.3 Utilidad de la interrupción int86 en C. 12
3.4 ¿Para qué sirve la función kbhit? 12
3.5 Equivalente de kbhit en otros lenguajes de programación. 12
4. Conclusiones. 12
5. Referencias. 14

Tabla de Imágenes.

Figura 1. Modelo de Sistema Monolítico. 5
Figura 2. Modelo cliente-servidor. 6
Figura 3. Máquina Virtual. 6
Figura 4. Sistema de capas. 7

1. Modelo de Sistemas Operativos.

Los Sistemas Operativos han evolucionado a lo largo de los años, surgiendo diferentes estructuras dependiendo de las necesidades de los usuarios. Los cambios y mejoras no se han dado únicamente por la experimentación con los SO, sino que también han sido provocadas por desarrollos en el hardware de las computadoras, además de las novedosas aplicaciones y los hilos de seguridad que hay en los softwares actuales. Algunas de las mejorías en el hardware son los sistemas de multiprocesadores, procesadores de alta potencia y velocidad, redes de mayor velocidad o la gran capacidad de memoria que tienen actualmente los dispositivos.

Estos cambios en las áreas mencionadas requieren tanto modificaciones en la arquitectura, como una reorganización en los Sistemas Operativos. Para ello se han distinguido algunas estructuras comunes que, a pesar de haber una amplia gama de diseños, la mayoría se acoplan a las características de las que se mencionarán a continuación.

1.1 Sistema monolítico.

Una gran parte de los Sistemas Operativos se caracterizan por ser monolíticos. Se basa en la ejecución del software en modo kernel. El SO es redactado como un conjunto de procedimientos que tienen alguna relación entre sí, todo en un programa binario con la capacidad de ser ejecutado. Al ser un solo programa que contiene todas las características del SO, es muy extenso. Debido a las características del modo kernel, cualquier proceso tiene la capacidad de llamar a cualquier otro en caso de necesitarlo. Son sistemas complejos ya que no hay limitantes en la comunicación entre procesos, por ende, puede tornarse complejo de manejar y de distinguir lo que está ocurriendo.

Es necesario compilar todos los procesos individuales a la hora de optar por un sistema como éste, para posteriormente relacionarlos y crear un único ejecutable, empleando un enlazador del sistema. Todos los procedimientos pueden ser vistos por parte de cualquier otro proceso. No obstante, y aunque parezca que no, este tipo de sistemas tienen estructura. Es necesario posicionar los parámetros en el lugar adecuado para optar por pedir los servicios propuestos por el SO. Estos parámetros esperan a la ejecución de una instrucción de trap (interrupción causada), necesaria para permutar entre el modo usuario y el modo kernel, obteniendo ahora el sistema operativo el control.

El SO es el encargado de analizar los parámetros, para posteriormente determinar la función o llamada al sistema que se llevará a cabo, teniendo así una clara estructura: programa principal que llama al procedimiento de servicio solicitado, conjunto de procesos de servicio que a su vez llaman al sistema, otro conjunto de procesos utilitarios que ayudan a los procesos de servicios. Algunos SO tienen apartados extras, como los drivers, que son necesarios para la utilización de dispositivos E/S y sistemas de archivos.

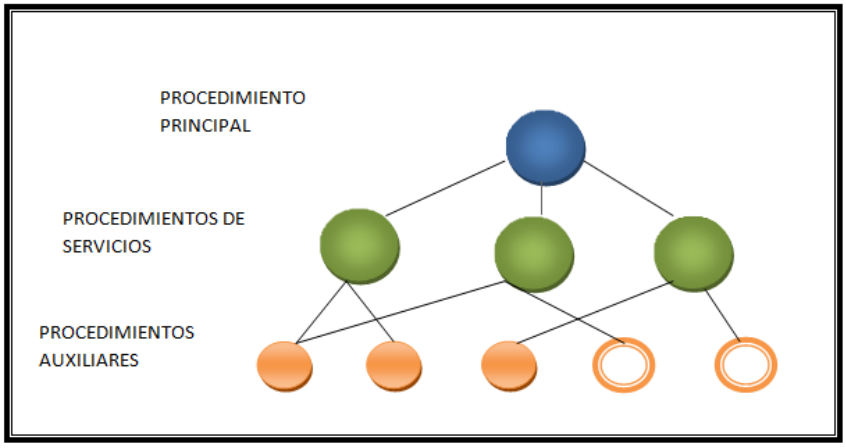

Figura 1. Modelo de Sistema Monolítico.

1.2 Modelo Cliente-Servidor.

Una de las bases del sistema monolítico es la importante presencial del núcleo. No obstante, surgió una tendencia en la que se buscaba que el núcleo sea mínimo, separándolo del código del software. Este tipo de sistema clasifica los procesos que puede haber, entre ellos están los servidores, que son los que proporcionan los servicios, y los clientes, que son los que emplean los servicios. El proceso realizado por el usuario, llamado proceso de cliente llama a un proceso servidor, para posteriormente realizar un trabajo y obtener una respuesta, limitando al núcleo a ser el gestor de esta comunicación entre clientes y servidores.

Los mensajes son el medio de comunicación más común. Este tipo de modelo suele diferenciar incluso hasta los equipos que usan los clientes y los servidores. Es decir, ambos procesos suelen ser ejecutados en distintos dispositivos, siendo conectados por alguna LAN o WAN. Este tipo de sistemas simplifica la ejecución, además de permitir utilizar de manera eficiente los recursos de la máquina. Este tipo de modelo se ha desarrollado tanto que, en la actualidad, los clientes se comunican con el servidor de manera gráfica, además de tener la posibilidad de interactuar con procesos auxiliares que son los encargados de hacer todo el proceso de la comunicación.

Es un modelo de alta fiabilidad, ya que cada servidor se encuentra en diferentes núcleos con una partición de memoria específica, protegiéndose así de otros servidores o clientes. Además, proporciona bases sustentables para la distribución de programas, pudiendo realizar este intercambio de mensajes sin encontrarse físicamente cerca. No obstante, tiene algunos aspectos negativos, como que se necesita una excelente infraestructura de comunicaciones, lo que puede tornarse costoso.

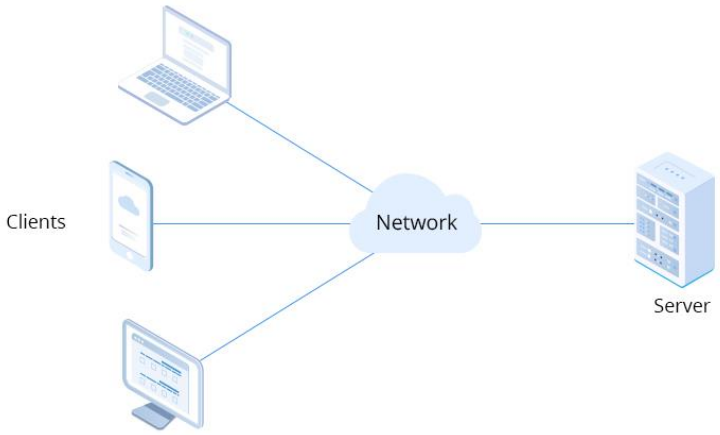

Figura 2. Modelo cliente-servidor.

1.3 Máquina virtual.

Una máquina virtual es un tipo de SO el cual se basa en comodidad gráfica para cada proceso, esto quiere decir que cada procedimiento posee una interfaz, simulando una máquina real. En una máquina virtual están separados la multiprogramación y la máquina extendida, conceptos que suelen ir de la mano. Las máquinas virtuales buscan combinar distintos SO, simulando ser máquinas distintas.

Estos Sistemas Operativos llaman monitor virtual a su núcleo, que es el agente encargado de desarrollar la multiprogramación, capaz de proporcionar las máquinas virtuales que los niveles de gestión superiores requieran. Al ser una simulación de máquinas, cada una de ellas puede ejecutar un SO diferente, siendo éste el que pueda controlar el usuario, otorgando flexibilidad y la posibilidad de implementar varios sistemas operativos en cada máquina virtual. A pesar de ello, cada SO es disjunto con los otros, por ende, es realmente complicado establecer alguna interacción o comunicación entre ellos.

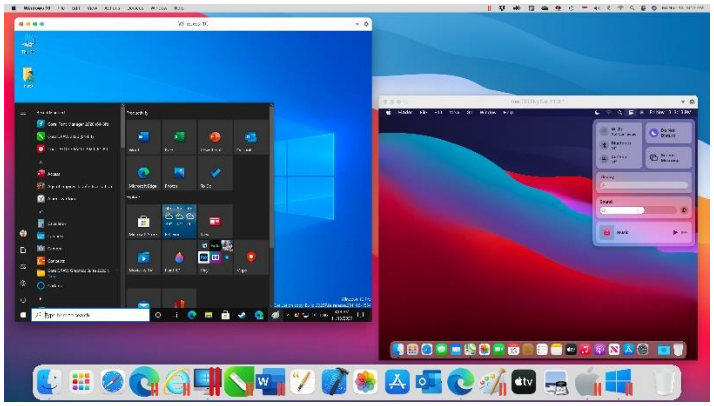

Figura 3. Máquina Virtual.

1.4 Capas.

La idea de ver un SO como capaz es determinar una cierta jerarquía, tomando como cimiento las capas inferiores, sirviendo en el funcionamiento de las superiores. “THE” fue el primer sistema de estas características, desarrollado por E. W. Dijkstra y sus estudiantes. Dicho sistema se basaba en el procesamiento por lotes, trabajando para una computadora holandesa. Este sistema poseía 6 capas o niveles. El nivel 0 indicaba al procesador el momento de permuta entre procesos. A partir de los siguientes niveles los procesos se tornaban secuenciales, teniendo la certeza de que se podía programar a pesar de estar ocurriendo otros procesos en el mismo procesador.

El siguiente nivel era el administrador de la memoria, determinando el espacio que tendría cada proceso en la memoria principal. La capa 2 proporcionaba la comunicación entre cada proceso y la consola o usuario. La capa posterior tenía el deber de organizar los dispositivos de entradas y salidas, además de guardar en búferes los flujos de información. La cuarta capa poseía los programas del usuario, y en la quinta se encontraba el proceso operador del sistema. Cada nivel tenía la capacidad de emplear información y procesos realizados por capas previas.

Esto derivó en lo que hoy conocemos como niveles de gestión, donde está el nivel de gestión del procesador, gestión de memoria, gestión de procesos que es el nivel encargado de la comunicación, el nivel de gestión de dispositivos E/S, y el nivel de gestión de la información. MULTICS fue otro sistema que empleó este tipo de SO, describiéndolo como anillos que compartían un centro. Cuando un nivel superior requería algún proceso de un nivel inferior requería de una llamada al sistema. Fueron los avances de hardware los que hicieron posible que se designaran procedimientos indivuales.

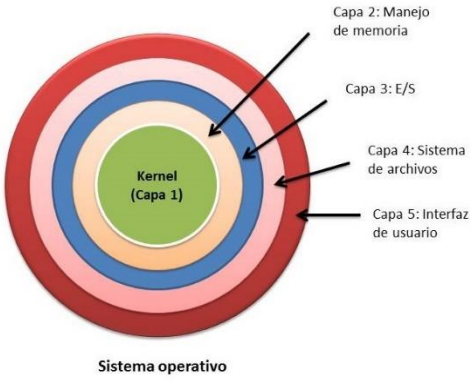

Figura 4. Sistema de capas.

1.5 Híbrido.

Un Sistema Operativo híbrido es aquel que combina características de varios modelos de SO, con la intención de tomar los mejores aspectos de cada uno de ellos y dependiendo de la aplicación que se le quiera dar. Aquellos sistemas que emplean la multiprogramación de tiempo

compartido y la de tiempo real son claros ejemplos de sistemas híbridos. Los sistemas de tiempo compartido permiten que los recursos del sistema estén disponibles para cada usuario, realizando una petición a la computadora que posteriormente es procesada y proporciona una respuesta en la terminal. Se entiende por recursos el procesador, memoria, dispositivos de entrada y salida, entre otros.

Por otro lado, un sistema de tiempo real es aquel que toma muy en cuenta el tiempo en que se tarda el obtener un resultado, además de la precisión de éste. Son sistemas que se basan en los procesos más que en los usuarios. Suelen ser utilizados en proyectos donde se maneja una gran cantidad de eventos que requieren ser procesados.

Un sistema híbrido es aquel que intenta tomar características de ambos tipos de modelos, tratando de complementar las ventajas y desventajas de cada uno. Los sistemas híbridos se han vuelto muy comunes en tiempos recientes, esto debido a que utiliza las ventajas de los modelos combinados, proporcionando mejor desempeño, mejoras en la fiabilidad, mayor flexibilidad- Bi obstante, un sistema híbrido es más complejo, siendo más demandante a la hora de administrar y mantener. Además, debido a su complejidad, son más costosos que un sistema de un único tipo, y podría tener problemas de compatibilidad. El reto es encontrar qué modelos se deben combinar y a dónde se quiere llegar con ellos.

2. Sistema Operativo macOS.

Mac Os es un Sistema Operativo nativo para la empresa Apple Inc., ésta dio a luz en el año 1976, gracias a sus fundadores Steve Jobs, Steve Wozniak y Ronald Wayne. El sistema mac OS actual está basado en Unix, utilizando los cimientos de una tecnología desarrollada por NeXT desde 1980, hasta que Apple la compró en 1997. Con el transcurrir de los años Mac OS X ha introducido muchas nuevas características, como la multitarea o una interfaz gráfica llamada Aqua.

Kodiak fue el nombre de la primear versión de Mac OS, aunque otros sistemas habían sido relacionados con la empresa. Esta versión fue publicada en otoño del año 2000 gracias a toda la retroalimentación que había obtenido de sus clientes de sus productos previos. No obstante, Kodiak duró menos de un año, ya que en primavera del año 2001 la versión 10.0, también conocida como Cheetah, fue publicada, desplazando así a Kodiak a un segundo plano.

La intención con mac OS era mejorar los sistemas operativos clásicos que habían sido empleados en las primeras Macintosh, desarrolladas en los 80’s. La publicación de Cheetah fue un precursor de la tendencia de Apple de nombrar a sus SO como felinos por casi una década, cuando la versión 10.9 fue nombrada Mavericks. Apple también se ha preocupado por equipar su sistema operativo para que sea lo más amigable posible para sus usuarios, realizando cambios como la integración de la tienda de aplicaciones App Store en 2008, inclusión de su sistema de nube llamado iCloud en 2011 para sincronizar los datos de los usuarios, introducción de una API de nuevos gráficos que tiene excelente rendimiento en juegos, modo oscuro en su versión

Mojave en 2018, entre otras. La versión más actual de macOS es macOS Ventura, estando en la versión 13.3 Beta 4.

2.1 Servicios que presta.

Si hay algo destacable en Apple es que se enfoca mucho en la experiencia de usuario, proporcionando distintos servicios que facilitan este aspecto. Se puede acceder a muchos servicios desde un menú de servicios o un menú contextual. Este SO tiene la intención de que la mayoría de los servicios que preste sean nativos de él, es decir, que sean desarrollados específicamente para este sistema operativo, esto con la intención de evitar que los usuarios tengan que ejecutar programas extras que pueden tornar todo tedioso.

Uno de los principales servicios que ofrece macOS es un menú de servicios. Esta funcionalidad permite al usuario utilizar procesos de otras aplicaciones para la tarea que se esté desempeñando en el momento, sin tener que abrir dichos programas. De igual forma el SO ofrece atajos de teclado, dándole incluso la posibilidad al usuario de crear sus propios atajos.

Apple también ofrece un automatizador, que permite definir tareas específicas que puedan ser realizadas sin la supervisión del usuario, ayudando a simplificar la realización de trabajos comunes. Siri es otro de los servicios más destacados de todos los dispositivos de la empresa, se trata de un asistente virtual capaz de realizar un sinfín de actividades, además de conectarse con otras aplicaciones e interactuar.

La búsqueda de archivos en una computadora puede tornarse complejo, es aquí donde Spotlight toma relevancia, proporcionando funcionalidades de búsqueda tanto de archivos como de otro tipo de información, ya sea contactos, emails, entre otros. Previniendo situaciones desgraciadas, MacOS ofrece Time Machine, un respaldado automático del dispositivo. AirDrop es uno de los mejores servicios del ecosistema de Apple, permite la transferencia de información de manera inalámbrica entre todos los dispositivos de la empresa. Siendo estas las principales funcionalidades y servicios, Apple ofrece muchos otros y sigue mejorando los ya existentes.

2.2 Objetivos.

Uno de los principales objetivos que busca Apple con sus SO y en especial con macOS es la longevidad. Es innegable que la empresa ofrece soporte y mantenimiento durante muchos años a sus dispositivos, por ende, el comprar un producto Apple es casi siempre sinónimo de adquirir una máquina potente que servirá durante mucho tiempo. Otras metas claras es emplear el poder que le proporciona UNIX con la simpleza de su laptop, la Mac, además de proveer movilidad y compatibilidad con las aplicaciones más importantes del mercado.

Como se ha mencionado previamente, la experiencia de usuario es de los principales puntos críticos para la empresa, teniendo como objetivo desarrollar un SO fácil de aprender y utilizar, además de que sea seguro, siempre se escuchan rumores de que los dispositivos Apple

no se infectan de malwares, y aunque no es cierto, sí tiene excelentes capas de seguridad. MacOS proporciona una plataforma abierta, que permite a los desarrolladores crear programas para sus dispositivos, aspecto que también ayuda al mantenimiento, siendo vital también para la creación del ecosistema, teniendo compatibilidad entre los sistemas operativos para tabletas y teléfonos.

2.3 Funciones.

MacOS, al estar basado en UNIX, interactúa con el hardware de sus dispositivos de manera básica, que permite ofrecer distintas funcionalidades además de los servicios mencionados. Iniciando por el hecho de la capacidad que tiene para correr aplicaciones variadas, capaces de adaptarse a las necesidades de cada usuario. MacOS, además, posibilita distintas herramientas para la administración y el manejo de archivos y carpetas, es uno de los SO más amigables para este rubro, entrando aquí el servicio de Spotlight mencionado con anterioridad. De igual forma posee una función de vista rápida de archivos, permitiendo una previsualización sin necesidad de abrirlo.

El sistema operativo provee distintas maneras de conexión a Internet, al igual que a dispositivos E/S, como puede ser una impresora, teniendo un servicio especial llamado AirPrinter, que permite realizar impresiones sin tener los drivers específicos de cada impresora. En su interior, macOS posee distintos programas para manejo de contenido multimedia, además de servicios de streaming. En todos los puntos positivos de Apple es mencionado AirDrop, ya que es de suma utilidad y una función excelente, además de tener excelentes recursos de apoyo y guías.

Las máquinas virtuales también pueden ser ejecutadas en macOS, permitiendo el uso de otros SO en un dispositivo Mac, funcionalidad importante para desarrolladores. Algunas Mac poseen una barra táctil en la cual pueden realizar algunos gestos, tales como ajustar el brillo y el volumen. Otra función muy interesante es la posibilidad de utilizar un iPad, la tablet desarrollada por Apple, como una segunda pantalla, además de la posibilidad de compartir archivos a través de estos dispositivos como si estuviesen conectados. Como podemos observar, las funciones que ofrece macOS son bastante amplias, permitiendo sacar el mayor provecho de sus dispositivos para así realizar trabajos más eficientemente.

2.4 Estructura.

Este Sistema Operativo nos ofrece una estructura por capas. Darwin es el nivel inferior, siendo el núcleo de Unix, concretamente en la distribución BSD. Es capaz de proporcionar los servicios necesarios para ejecutar los programas esenciales del dispositivo, tal como el sistema de archivos, conexión de redes y seguridad. El sistema gráfico es la siguiente capa, empleando framework como Quartz, OpenGL y QuickTime. Para el nivel de los programas se emplean componentes como Classic, Carbon, Cocoa y Java, para finalizar con la interfaz de usuario llamada Aqua.

Este sistema no fue creado de un día para otro, y ha sido un proceso constante de mejoría desde los primeros modelos hasta el actual. MacOS no deja de tener el kernel, que se encarga de la administración de los recursos tanto de memoria como de procesador, además de proveer las bases para la ejecución de aplicaciones. Además, es un sistema enfocado en multiusuario, lo que quiere decir que personas diferentes pueden crear sesiones en una Mac, y cada sesión tendrá características y configuraciones diferentes. La base de la estructura proporciona los aspectos básicos para cada usuario.

3. Preguntas.

3.1 ¿Qué significa JCL?

JCL hace referencia a Job Control Language. El monitor requiere de instrucciones, y el JCL es un lenguaje especial que se encarga de proporcionar estos pasos. Es sumamente usado en sistemas de procesamiento por lote, además de en subsistemas, ya que tiene la capacidad de determinar cuáles procesos ejecutar, además de demandar algún archivo o dispositivo que se requiera para la realización de una tarea, e incluso puede determinar si se debe omitir algún procedimiento.

Es fundamental ya que funge como mediador entre las aplicaciones requeridas por el usuario, el sistema operativo y el hardware del dispositivo. El surgimiento fue gracias a la abstracción de un trabajo, entendiendo a un job como pequeños pasos necesarios para la realización de alguna tarea, en este caso, ejecutar un programa.

3.2 Diferencia entre procesamiento por lotes y procesamiento por lotes con multiprogramación.

Primeramente, es importante destacar los aspectos por los que se relacionan estos modelos. El procesamiento por lotes y el procesamiento por lotes con multiprogramación tienen la labor de ejecutar los programas o trabajos en un Sistema Operativo. El procesamiento por lote simple se basa en la acumulación de trabajos de la misma índole, para que posteriormente, puedan empezar a ejecutarse sin la necesidad de vigilancia de un usuario, a modo de que cuando un trabajo termine, otro empiece a correrse y así sucesivamente.

Por otro lado, el procesamiento por lotes con multiprogramación permite, como su nombre lo dice, realizar múltiples procesos simultáneamente, sin necesidad de requerir varios dispositivos, ya que es posible realizarlo en la misma máquina. Los Sistemas Operativos de este tipo ejecutan las tareas necesarias para el usuario, vigilando atentamente el desempeño en sus actividades y teniendo la potestad de permutar entre procesos. Por sus características, el procesamiento por lotes simple es más lento, mientras que con multiprogramación permite un mejor aprovechamiento de los recursos de la máquina.

3.3 Utilidad de la interrupción int86 en C.

De manera resumida, la interrupción int86 en C facilita la interrupción de los servicios de la BIOS y del sistema operativo del disco, también conocido como DOS. Por ende, la principal utilidad es el poder interrumpir cualquiera de estos servicios. Otra utilidad bastante interesante del int86 es que puede ser usado para debuggear, esto debido a su habilidad de analizar e ingresar a registros y a la memoria, pudiendo así comprender la manera en la que se está ejecutando el programa que se quiere debuggear. No obstante, utilizar int86 para debug puede ser complicado, por ende, hay que saber usarlo correctamente, como, por ejemplo, para establecer puntos muertos en los códigos.

3.4 ¿Para qué sirve la función kbhit?

Es una función de tipo booleano, es decir, indica si algo es verdadero o falso. En este caso, kbhit se encarga de determinar si alguna tecla está siendo presionada, devolviendo verdadero en dicho caso. Es parte de la librería conio.h. Es de suma utilidad a la hora de realizar proyectos que requieran interactividad con dispositivos E/S, en este caso el teclado, para así tener acciones de respuestas a posibles gestos del usuario.

3.5 Equivalente de kbhit en otros lenguajes de programación.

En java, por ejemplo, existen varias maneras de detectar alguna acción con respecto a las teclas. Una de las maneras es utilizando “java.awt.event.KeyEvent”, está función proporciona la información de qué tecla se está presionando, y realizando ciertas modificaciones puede crearse una función que devuelva verdadero o falso si se presiona alguna tecla. Otra manera es con “System.in.read()”, que tampoco devuelve verdadero o falso, pero haciendo la función correcta puede tenerse un comportamiento similar a kbhit.

Python, en cambio, si posee una librearía que cuenta con una función de kbhit. En su caso, la librería se llama msvcrt, y la función se llama igualmente kbhit. Está funciona igual que en C, devolviendo 1 o 0 en caso de que se presione o no una tecla.

4. Conclusiones.

Uno de los aspectos que pude notar gracias a esta investigación es que el hecho de que haya competencia en el desarrollo de buenos sistemas operativos es lo que ha permitido que el nivel de desarrollo haya llegado a donde estamos hoy en día. Este proceso de mejora se ha basado en utilizar el conocimiento previo de manera estructural, por ejemplo, todos toman el concepto de kernel del sistema monolítico para algunas funcionalidades. A partir de ahí toman aspectos positivos de modelos previos y lo adaptan para realizar un mejor trabajo.

De los modelos más interesantes en mi opinión es el de cliente-servidor. Actualmente cursamos la materia de redes de computadoras y veo la trascendencia que tiene este tipo de abstracción, lo que consecuentemente ha permitido que nos comuniquemos al nivel en el que lo hacemos en la actualidad. Pero no solo se queda en aspectos de redes, sino que puede trasladarse a Sistemas Operativos. Una de las cosas que más destaqué acerca del sistema operativo investigado fue su ecosistema, y lo eficiente que es el hecho de poder compartir información de manera inalámbrica a través de sus dispositivos. Todo esto no sería posible sin la ayuda de modelos cliente-servidor. Las máquinas virtuales también es un concepto muy interesante, me quiero adentrar en él ya que me llama la atención del área de ciberseguridad, y las máquinas virtuales son el pan de cada día en ese tipo de temas.

Con respecto al SO, debo admitir que era de los que les alarmaba por los precios exagerados que tiene la empresa investigada, y aunque me siguen pareciendo extremadamente altos para el mercado actual, debo admitir que el nivel de SO que han logrado, además de la comunicación entre sus diferentes tipos de dispositivos es admirable y útil, creando así la fama tan merecida que tienen. Además, es enriquecedor saber que aún hay lenguajes y funciones que nos permiten adentrarnos y controlar procesos de niveles de gestiones inferiores. Con la erupción de lenguajes de tan alto nivel parece que no debemos preocuparnos por las cuestiones básicas, no obstante, siempre es bueno saberlo. Esto es lo que nos ofrece int86 y kbhit, que sabiendo como usarlos medianamente, pueden ayudarnos a realizar muchas tareas y a entender conceptos cruciales.

5. Referencias.

512 Pixels. (2017). *Apple's goals for the Mac operating system*. https://512pixels.net/2017/02/apples-goals-for-the-mac-operating-system/

Butler, S. (2021). What is the Services Menu in macOS. *macReports*. https://macreports.com/what-is-the-services-menu-in-macos/

Castro, K. (2020). *Mac OS X Structure*. Tutorialspoint. https://www.tutorialspoint.com/mac-os-x-structure

Flynn, I. M., & McHoes, A. M. (2001). *Sistemas operativos*. Cengage Learning Latin America.

GeeksforGeeks. (2022). *Kbhit in C language*. GeeksforGeeks. https://www.geeksforgeeks.org/kbhit-c-language/

La Red Martínez, D. L. (2004). Sistemas operativos. El Cid Editor. https://elibro-net.wdg.biblio.udg.mx:8443/es/lc/udg/titulos/77467

Obari, D. (2022). The coolest MacOS features we got in 2022. *MUO*. https://www.makeuseof.com/coolest-macos-features-in-2022/

Raya Cabrera, J. L. (2015). Implantación de sistemas operativos. RA-MA Editorial. https://elibro-net.wdg.biblio.udg.mx:8443/es/lc/udg/titulos/62453

Silberschatz, A., Galvin, P. B., & Gagne, G. (2012). *Operating system concepts*. Wiley.

Staff, H. C. (2023). *Complete history of Mac OS*. History-Computer. https://history-computer.com/complete-history-of-mac-os/

Stallings, W. (2015). *Operating systems: Internals and Design Principles*. Pearson Prentice Hall.

Tanenbaum, A. S. (2008). *Sistemas Operativos Modernos*. Pearson Prentice Hall.