



Instituto Politecnico Nacional

Escuela Superior de Cómputo



Grupo: 2CM12

Equipo 1

Integrantes :

Baldovinos Gutiérrez Kevin

Bocanegra Heziquio Yestlanezi

Castañares Torres Jorge Davidad

Hernández Hernández Ruth Esther

Profesor :

Jorge Cortes Galicia



UNIDAD 4

Dispositivos entrada/salida

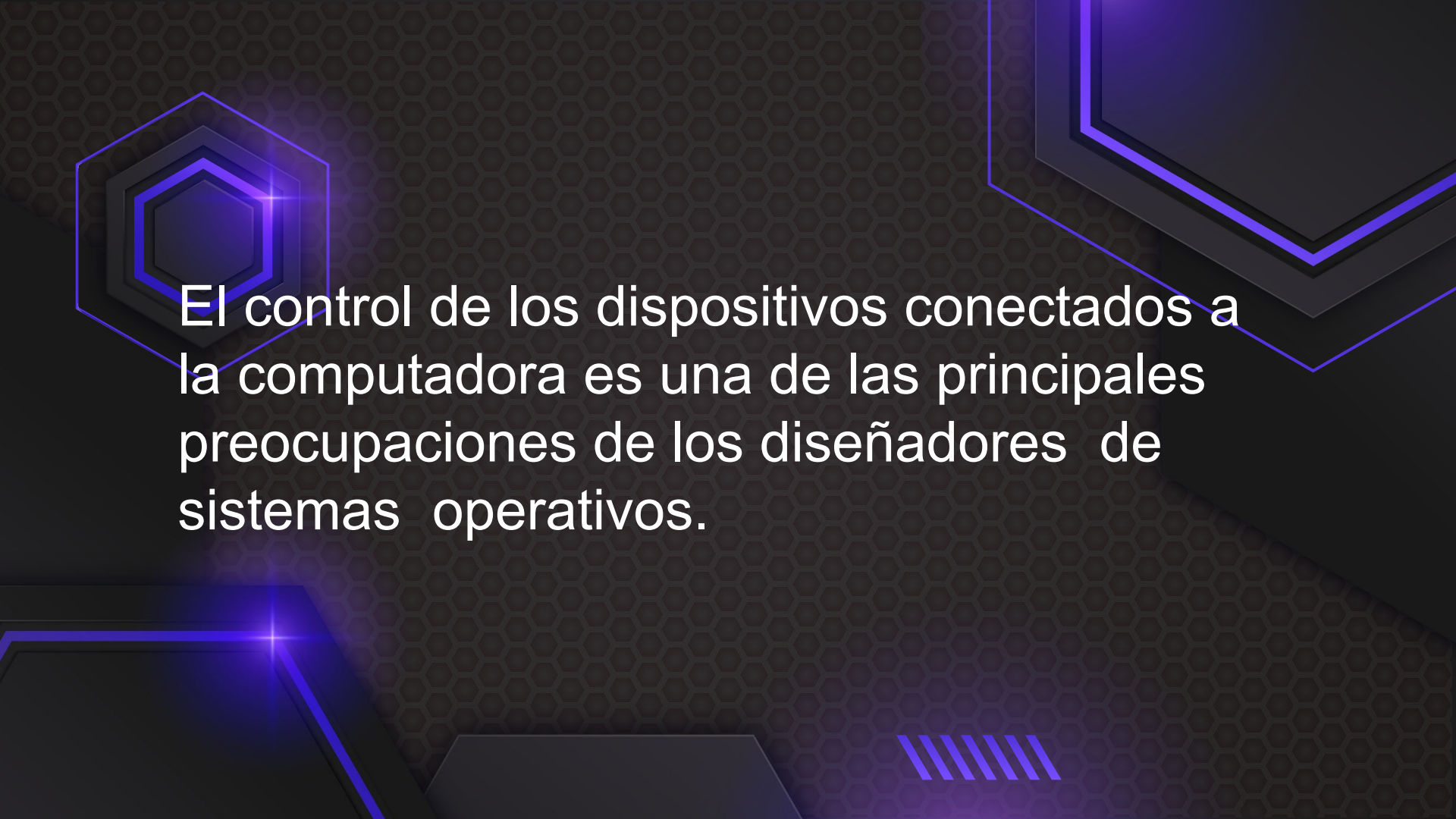
Equipo 1



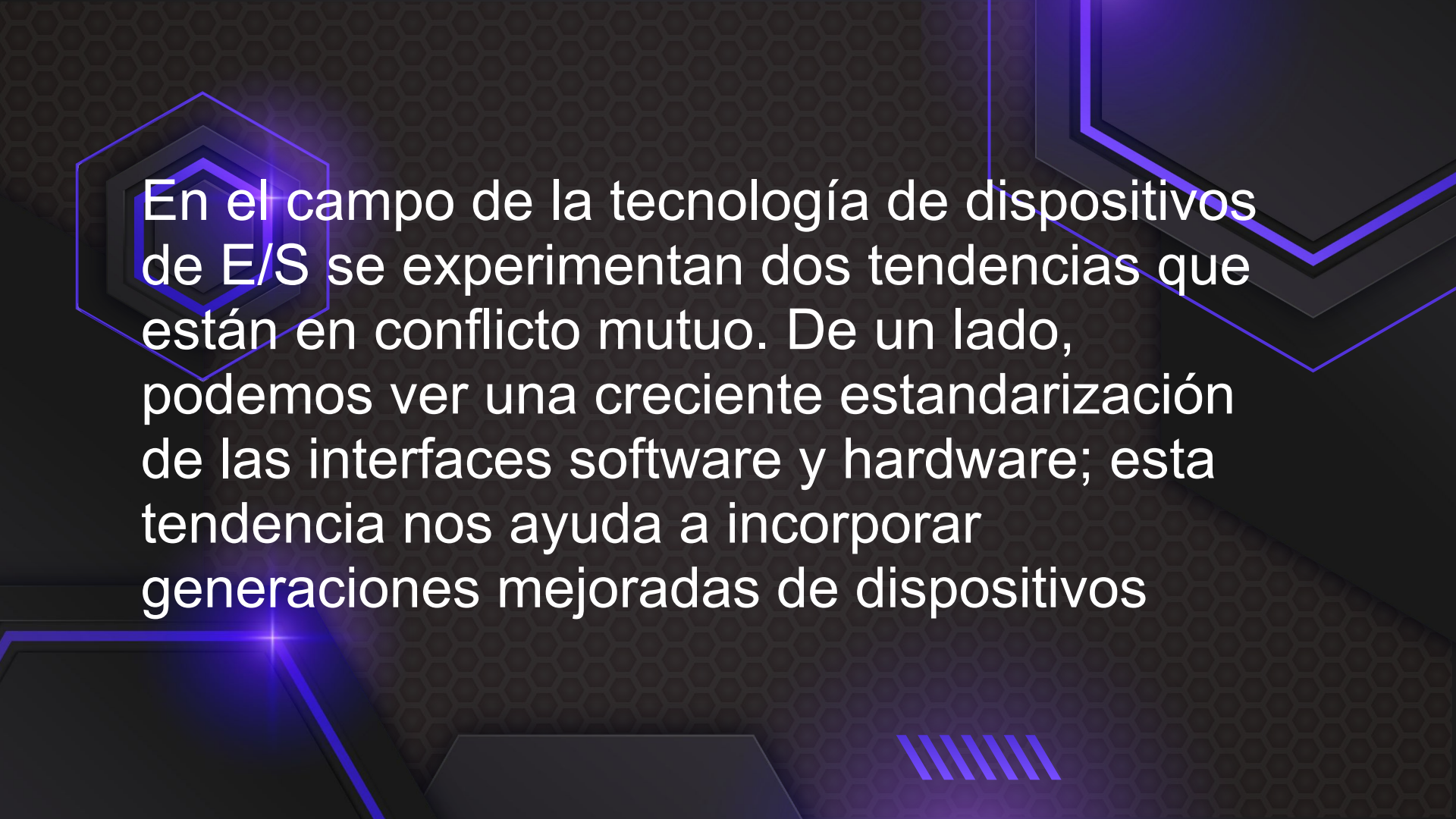
4.1 Principios del hardware de E/S

13.1 Introducción






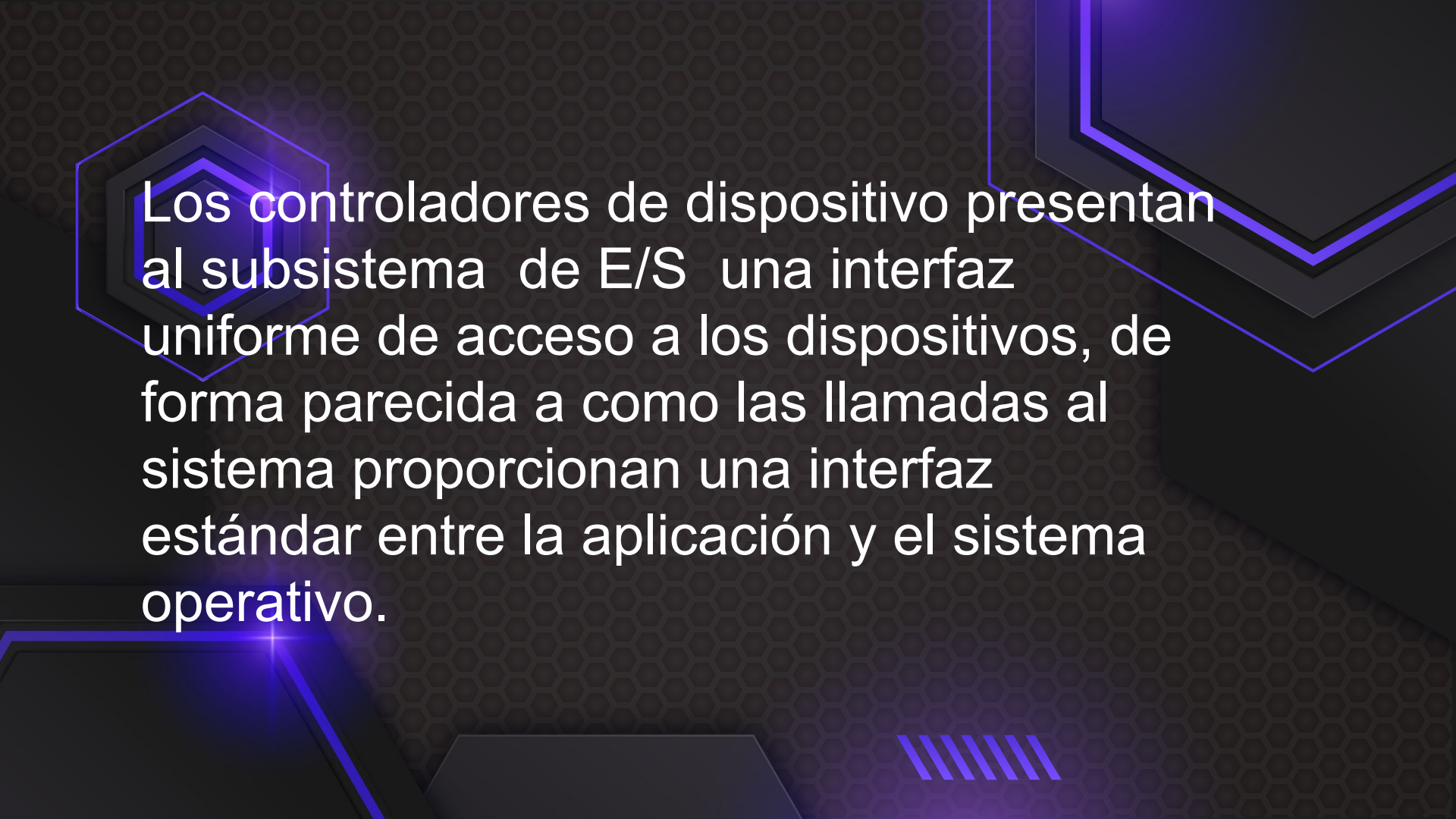
El control de los dispositivos conectados a la computadora es una de las principales preocupaciones de los diseñadores de sistemas operativos.




En el campo de la tecnología de dispositivos de E/S se experimentan dos tendencias que están en conflicto mutuo. De un lado, podemos ver una creciente estandarización de las interfaces software y hardware; esta tendencia nos ayuda a incorporar generaciones mejoradas de dispositivos



dentro de las computadoras de sistemas operativos existentes. Por otro lado, podemos ver variedad cada vez más amplia de dispositivos de E/S; algunos nuevos dispositivos son tan distintos de los dispositivos anteriores que constituye todo un desafío incorporarlos en las computadoras y los sistemas operativos.



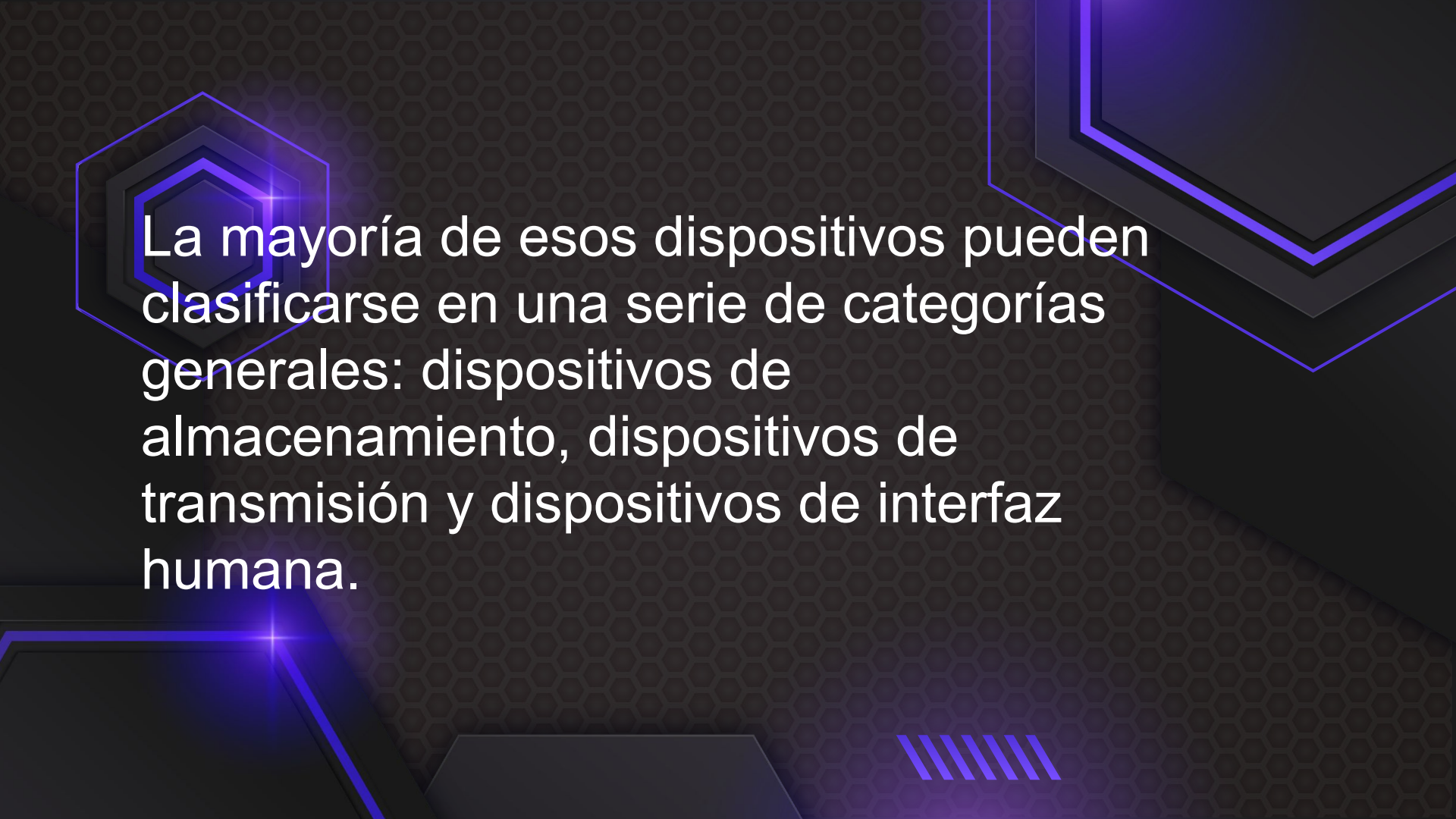
Los controladores de dispositivo presentan al subsistema de E/S una interfaz uniforme de acceso a los dispositivos, de forma parecida a como las llamadas al sistema proporcionan una interfaz estándar entre la aplicación y el sistema operativo.



13.2

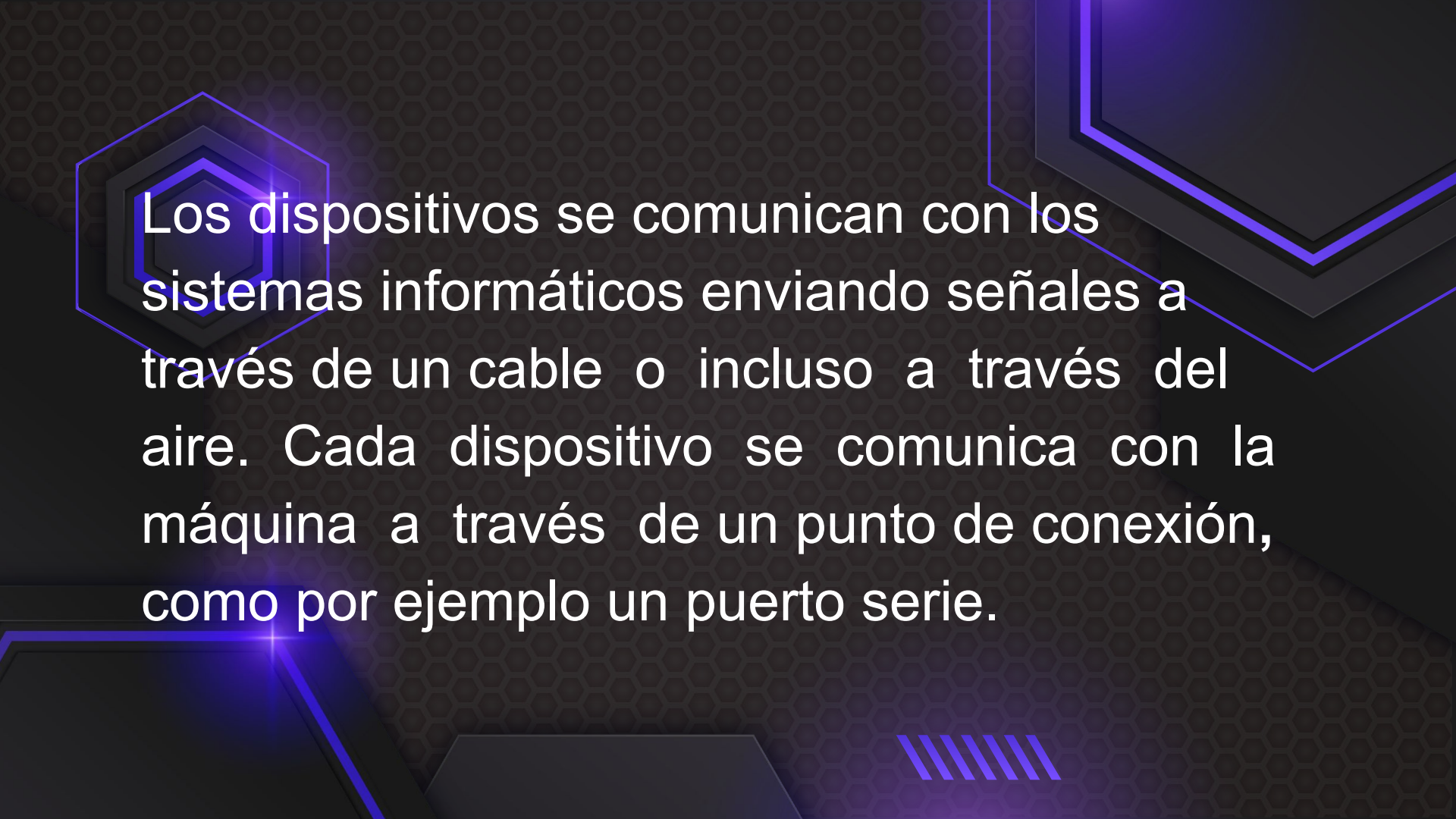
Hardware de E/S



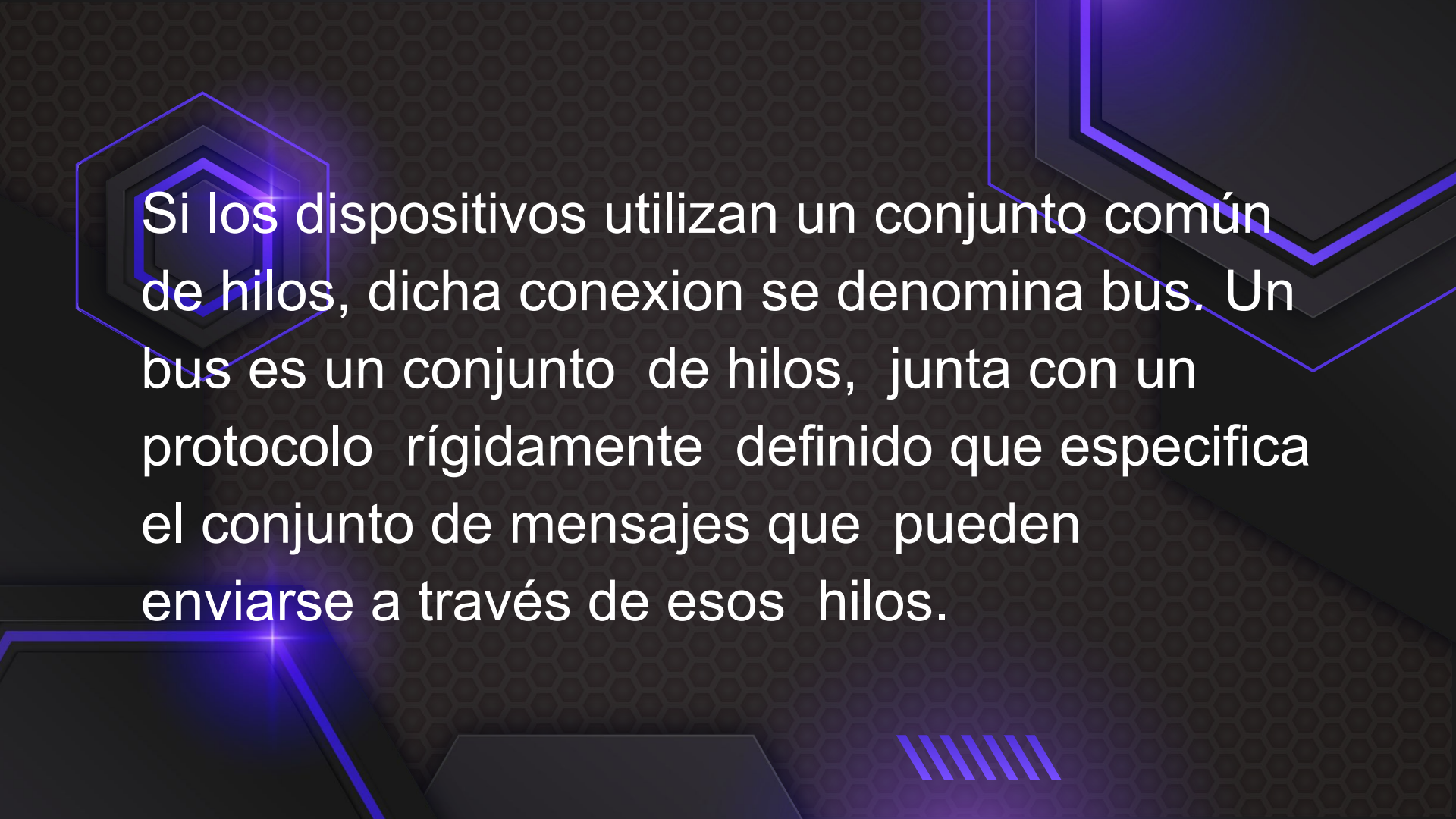


La mayoría de esos dispositivos pueden clasificarse en una serie de categorías generales: dispositivos de almacenamiento, dispositivos de transmisión y dispositivos de interfaz humana.






Los dispositivos se comunican con los sistemas informáticos enviando señales a través de un cable o incluso a través del aire. Cada dispositivo se comunica con la máquina a través de un punto de conexión, como por ejemplo un puerto serie.

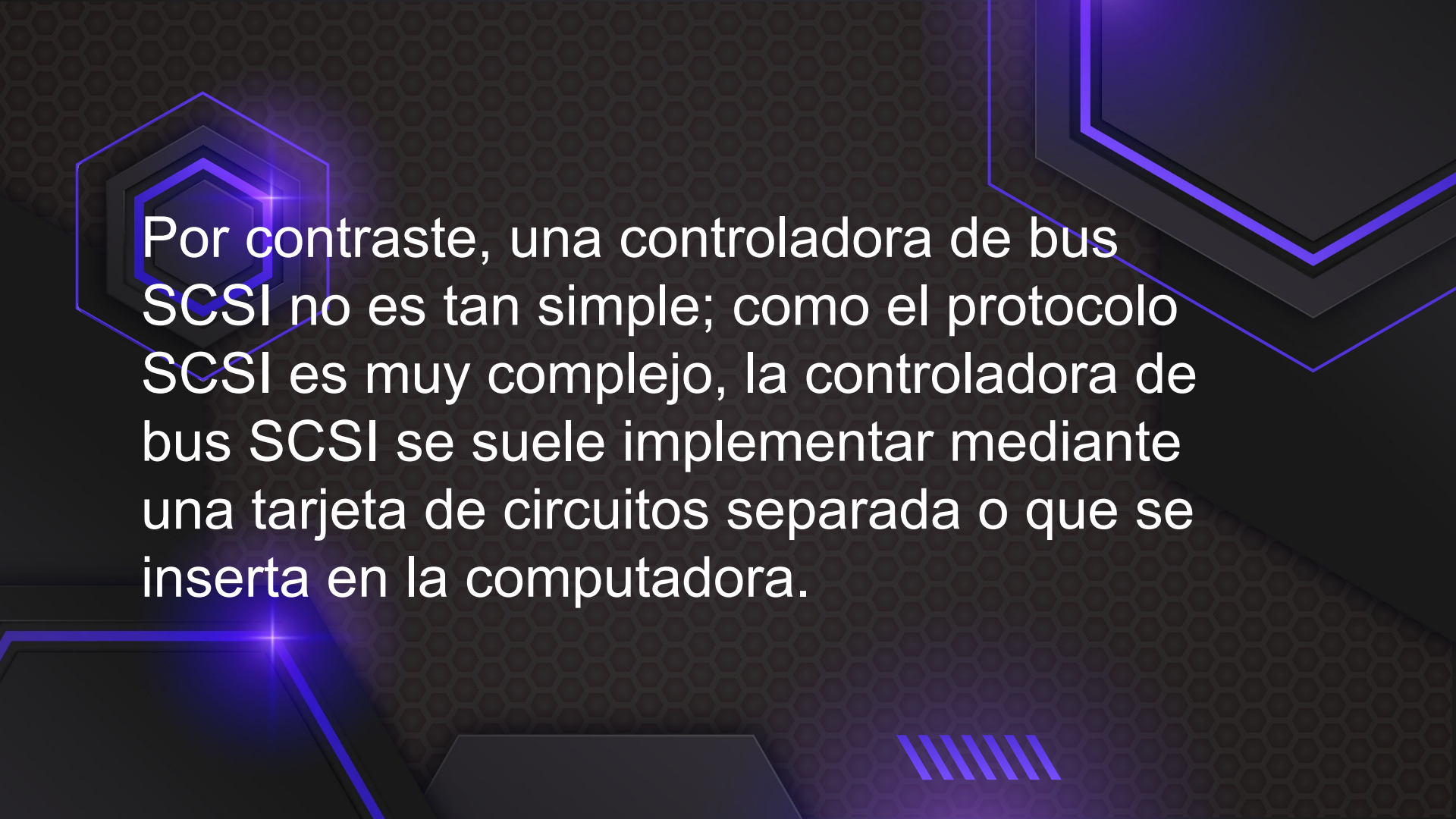


Si los dispositivos utilizan un conjunto común de hilos, dicha conexión se denomina bus. Un bus es un conjunto de hilos, junto con un protocolo rígidamente definido que especifica el conjunto de mensajes que pueden enviarse a través de esos hilos.



Cuando el dispositivo *A* tiene un cable que se inserta en el dispositivo *B*, y el dispositivo *B* tiene un cable que se inserta en el dispositivo *C*, y el dispositivo *C* se inserta en un puerto de la computadora, este tipo de dispositivo se denomina **conexión en cascada**. Las conexiones en cascada suelen funcionar como un bus.

Una **controladora** es una colección de componentes electrónicos que permite controlar un puerto, un bus o un dispositivo. Un ejemplo simple de controladora de dispositivo sería la controladora de un puerto serie: se trata de un único chip dentro de la controladora que controla las señales que se transmiten a través de los hilos de un puerto serie



Por contraste, una controladora de bus SCSI no es tan simple; como el protocolo SCSI es muy complejo, la controladora de bus SCSI se suele implementar mediante una tarjeta de circuitos separada o que se inserta en la computadora.

El procesador se comunica con la controladora leyendo y escribiendo patrones de bits en dichos registros. Una forma de llevar a cabo esta comunicación es utilizando instrucciones de E/S especiales que especifican la transferencia de un byte o de una palabra a una dirección de puerto de E/S. La instrucción de E/S configura las líneas de bus para seleccionar el dispositivo apropiado y para leer o escribir bits en un registro del dispositivo.



Alternativamente, la controladora de dispositivo puede soportar una E/S mapeada en **memoria**. En este caso, los registros de control de! dispositivo están mapeados en el espacio de direcciones del procesador. La CPU ejecuta las solicitudes utilizando instrucciones estándar de transferencia de datos para leer y escribir los registros de control del dispositivo.

Los registros de datos tienen normalmente entre 1 y 4 bytes de tamaño. Algunas controladoras tienen chips FIFO que permiten almacenar varios bytes de datos de entrada y de salida, para expandir la capacidad de la controladora más allá del tamaño que el registro de datos tenga. Un chip FIFO puede almacenar una pequeña rafaga de datos hasta que el dispositivo o *host* sea capaz de recibir dichos datos.

13.2.1

Sondeo



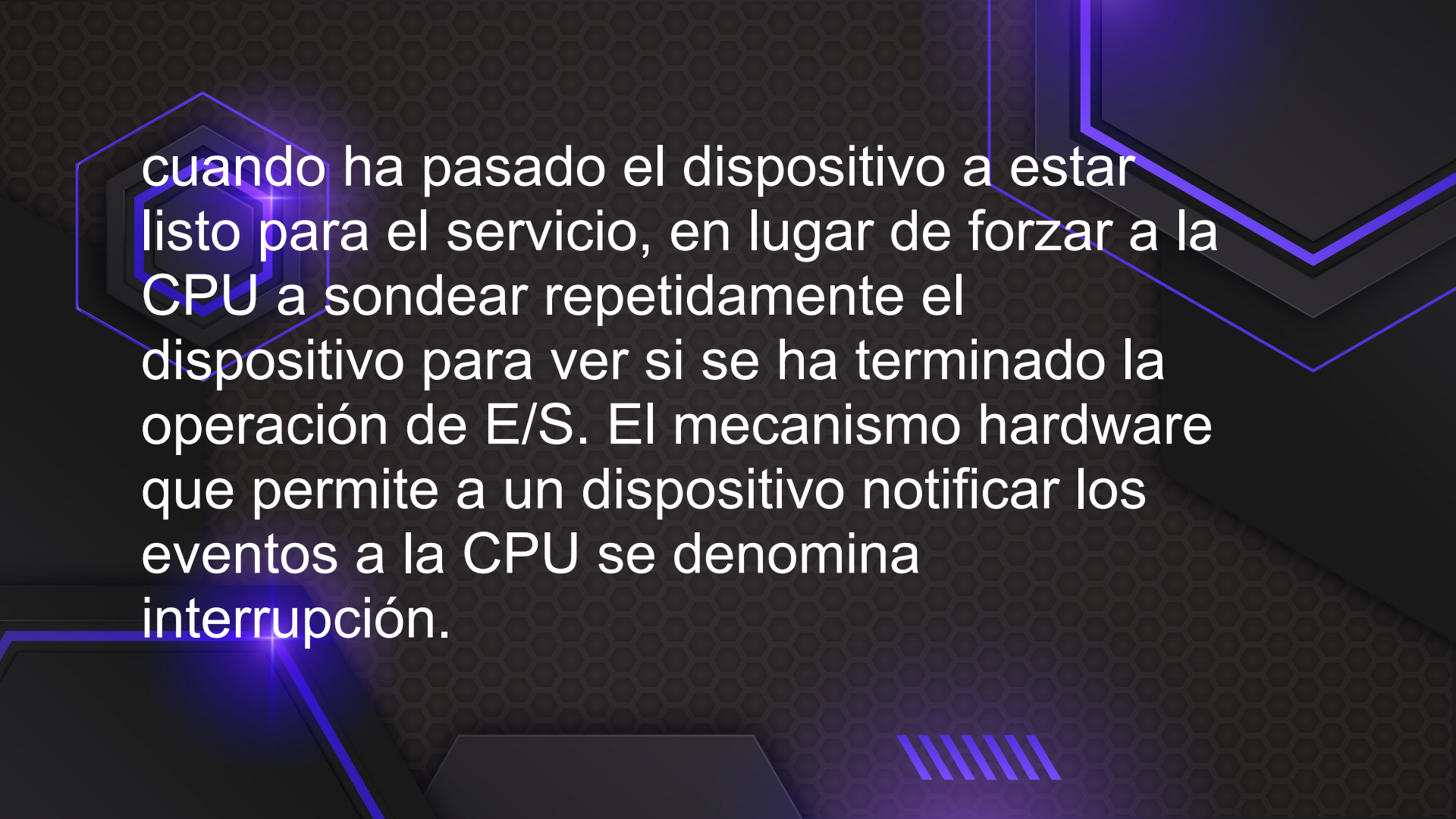
El protocolo completo de interacción entre el *host* y una controladora puede ser intrincado, pero la noción básica de negociación resulta muy simple. Vamos a explicar el concepto de negociación mediante un ejemplo. Supongamos que se utilizan 2 bits para coordinar la relación productor consumidor entre la controladora y el *host*. La controladora indica su estado mediante el bit de *ocupado* en el registro de *estado*.

La controladora activa el bit de *ocupado* cuando está ocupada trabajando y borra bit de *ocupado* cuando está lista para aceptar el siguiente comando. El *host* indica sus deseos mediante el bit de *comando preparado* en el registro de *comando*: el *host* activa el bit de *comando preparado* cuando hay disponible un comando para que la controladora lo ejecute.




En muchas arquitecturas informáticas, para sondear un dispositivo basta con tres ciclos de instrucciones de CPU: *leer* un registro del dispositivo, efectuar una operación de *and Lógica* para extraer un bit de estado y *saltar* si ese bit es distinto de cero. Obviamente, la operación básica de sondeo resulta muy eficiente.

Pero el sondeo pasa a ser ineficiente cuando se le ejecuta de manera repetida para encontrar únicamente que solo en raras ocasiones está listo el dispositivo para ser servido, mientras otras tareas útiles de procesamiento que podría haber ejecutado la CPU permanecen sin hacer. En tales casos, puede ser más eficiente que la controladora hardware notifique a la CPU



cuando ha pasado el dispositivo a estar listo para el servicio, en lugar de forzar a la CPU a sondear repetidamente el dispositivo para ver si se ha terminado la operación de E/S. El mecanismo hardware que permite a un dispositivo notificar los eventos a la CPU se denomina interrupción.



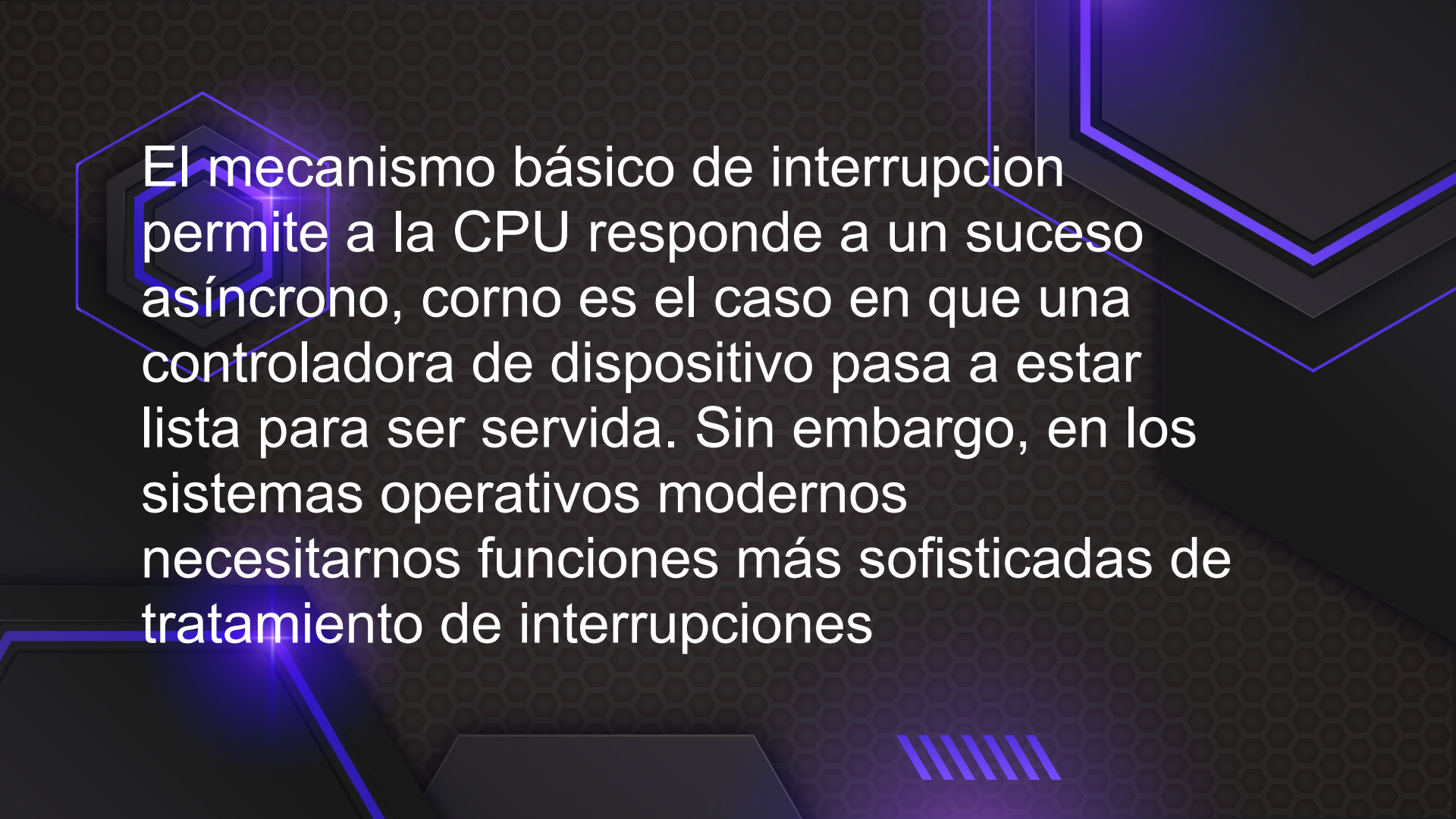
13.2.2

Interrupciones



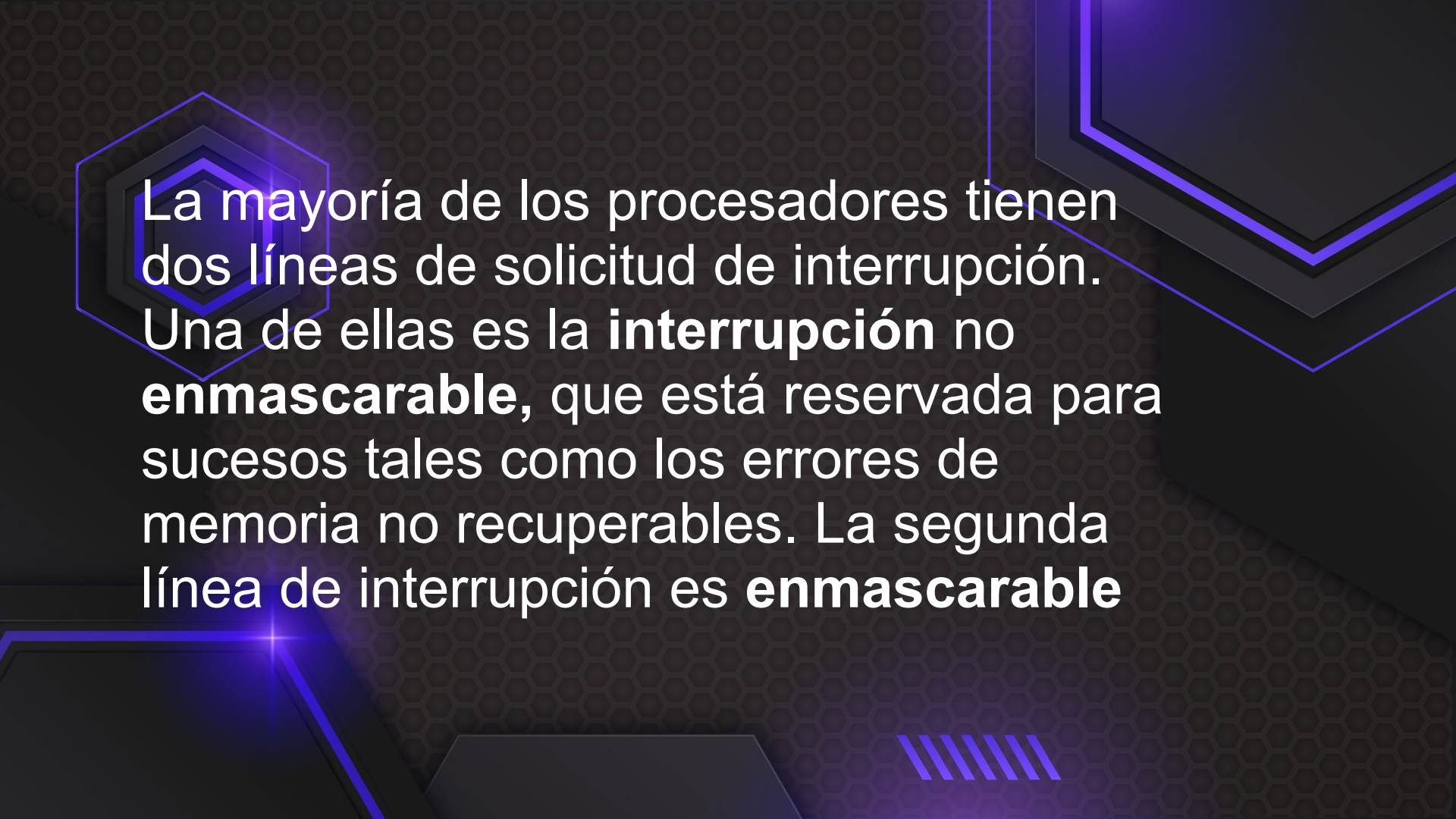
El mecanismo básico de interrupción funciona de la forma siguiente. El hardware de la CPU tiene un hilo denominado **línea de solicitud de interrupción** que la CPU comprueba después de ejecutar cada instrucción. Cuando la CPU detecta que una controladora ha activado una señal a través de la línea de solicitud de interrupción, la CPU guarda el estado actual y salta a la **rutina de tratamiento de interrupciones** situada en una dirección fija de la memoria.






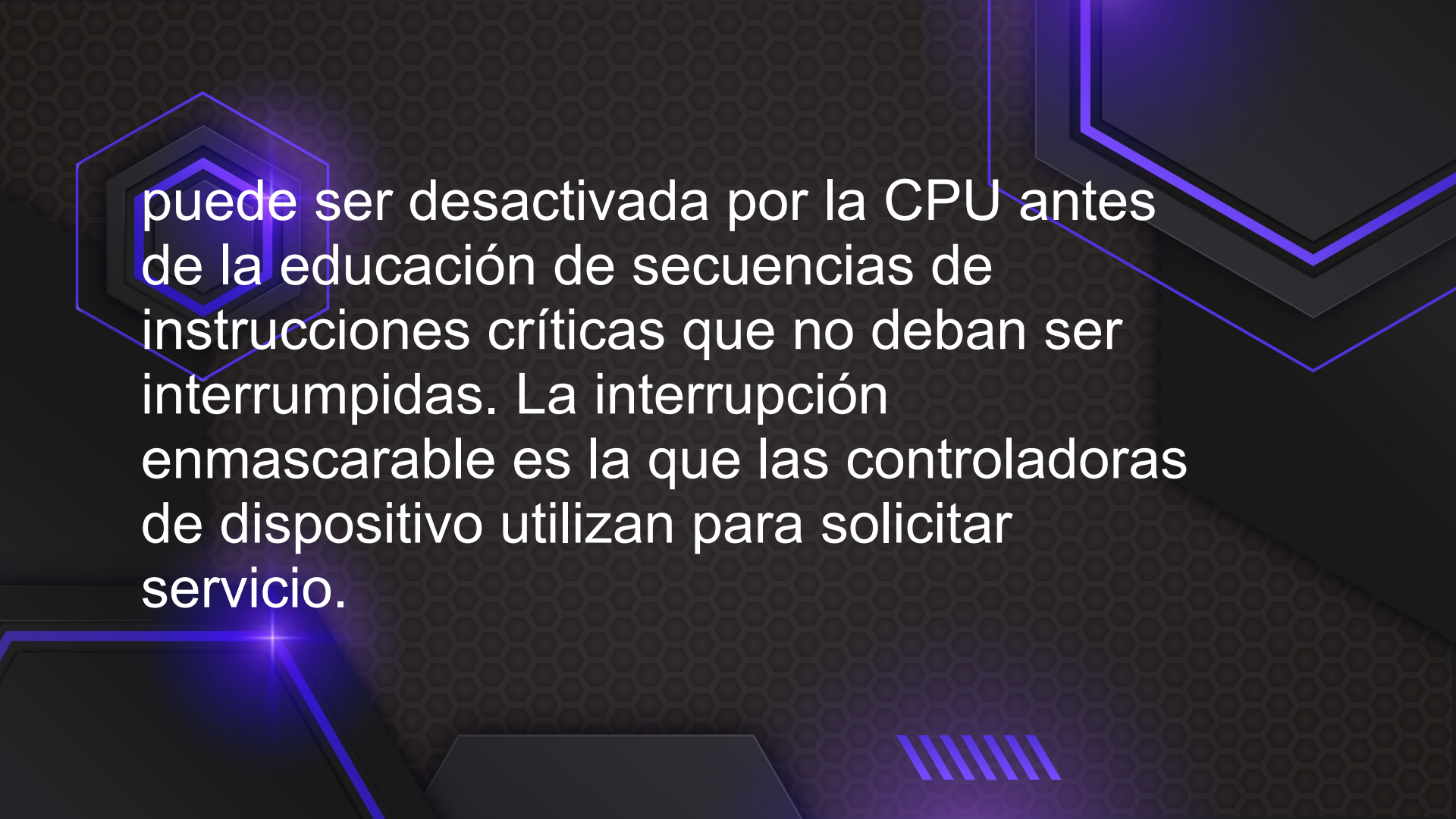
El mecanismo básico de interrupción permite a la CPU responder a un suceso asíncrono, como es el caso en que una controladora de dispositivo pasa a estar lista para ser servida. Sin embargo, en los sistemas operativos modernos necesitamos funciones más sofisticadas de tratamiento de interrupciones





La mayoría de los procesadores tienen dos líneas de solicitud de interrupción. Una de ellas es la **interrupción no enmascarable**, que está reservada para sucesos tales como los errores de memoria no recuperables. La segunda línea de interrupción es **enmascarable**





puede ser desactivada por la CPU antes de la ejecución de secuencias de instrucciones críticas que no deban ser interrumpidas. La interrupción enmascarable es la que las controladoras de dispositivo utilizan para solicitar servicio.



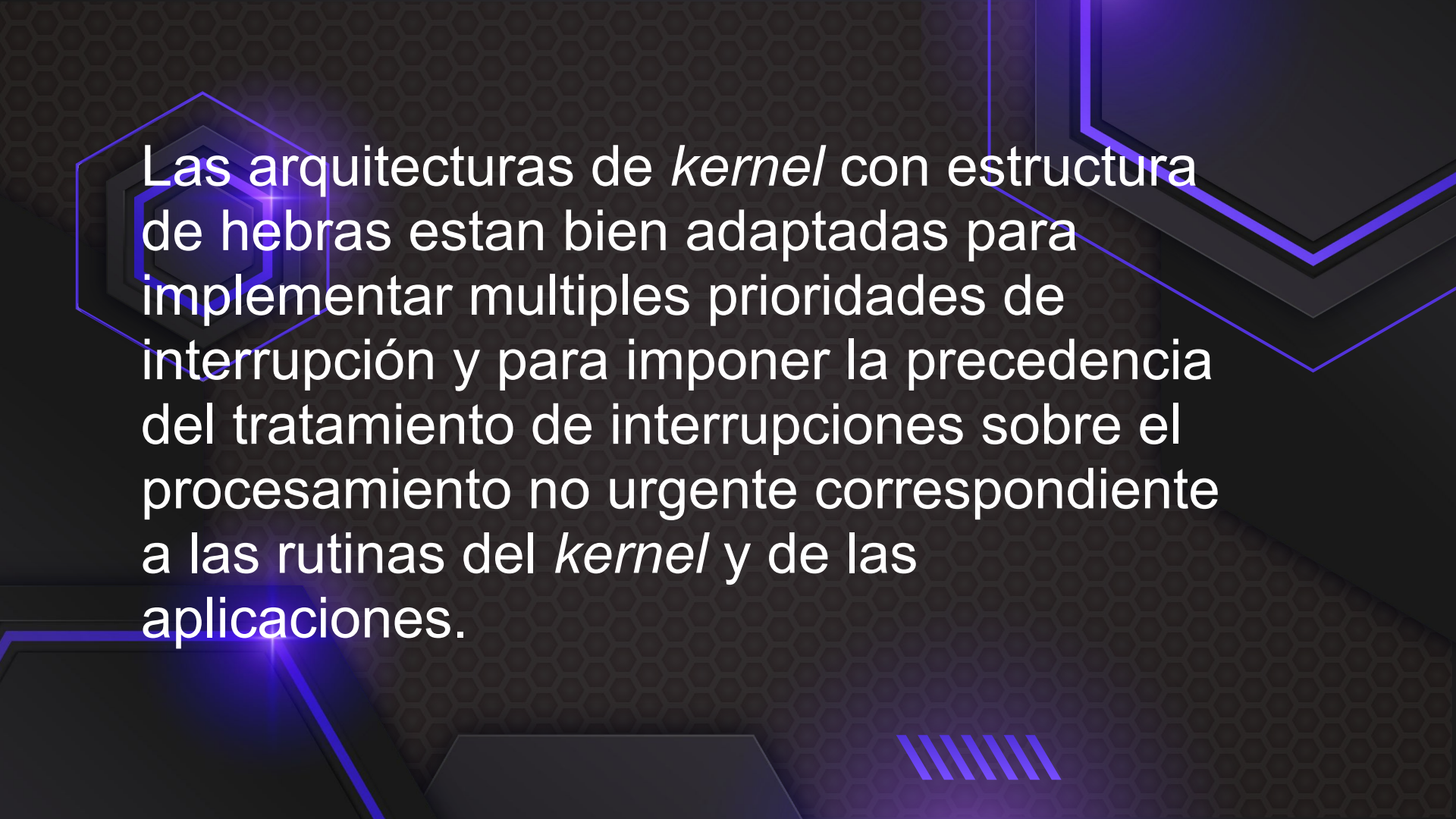
El mecanismo de interrupción acepta una **dirección**, que es el número que selecciona una rutina específica de tratamiento de interrupciones de entre un pequeño conjunto de rutinas disponibles. En la mayoría de las arquitecturas, esta dirección es un desplazamiento dentro de una tabla denominada **vector de interrupciones**

El mecanismo de interrupciones también implementa un sistema de niveles de prioridad de **interrupción**. Este mecanismo permite a la CPU diferir el tratamiento de las interrupciones de baja prioridad sin enmascarar todas las interrupciones, y hace posible que una interrupción de alta prioridad desaloje a otra interrupción de prioridad más baja.

El mecanismo de interrupciones se utiliza también para gestionar una amplia variedad de **excepciones**, como la división por cero, el acceso a direcciones de memoria protegidas o no existentes, o el intento de ejecutar una instrucción privilegiada en modo usuario. Los sucesos que generan interrupciones tienen una propiedad en común: son sucesos que inducen a la CPU a ejecutar una rutina urgente y autocontenido

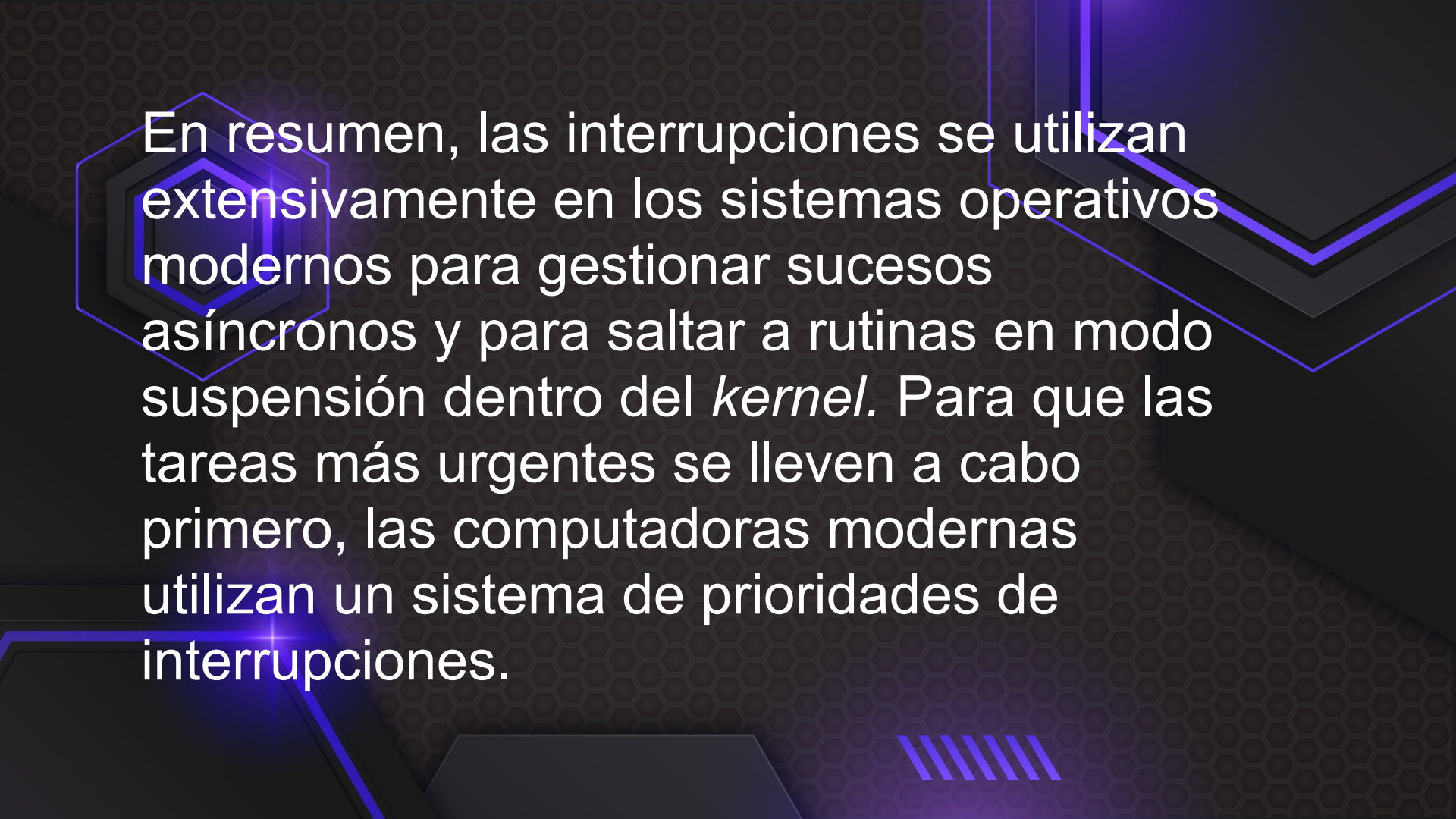
Otro ejemplo es el de la implementación de las llamadas al sistema. Usualmente, los programas utilizan llamadas a biblioteca para realizar llamadas *al sistema* . Estas rutinas de biblioteca comprueban los argumentos proporcionados por la aplicación, construyen una estructura de datos para entregar esos argumentos *al kernel* y luego ejecuta una instrucción especial denominada **interrupción software**.

Las interrupciones tambien pueden usarse para gestionar el flujo de control dentro del *kernel*. Por ejemplo, considere el procesamiento requerido para completar una lectura de disco. Uno de los pasos es copiar los datos desde el espacio del *kernel* al buffer de usuario; esta operación de copia lleva mucho tiempo pero no es urgente, así que no debe bloquear el tratamiento de otras interrupciones de alta prioridad.




Las arquitecturas de *kernel* con estructura de hebras estan bien adaptadas para implementar multiples prioridades de interrupción y para imponer la precedencia del tratamiento de interrupciones sobre el procesamiento no urgente correspondiente a las rutinas del *kernel* y de las aplicaciones.





En resumen, las interrupciones se utilizan extensivamente en los sistemas operativos modernos para gestionar sucesos asíncronos y para saltar a rutinas en modo suspensión dentro del *kernel*. Para que las tareas más urgentes se lleven a cabo primero, las computadoras modernas utilizan un sistema de prioridades de interrupciones.



Las controladoras de dispositivo, los fallos hardware y las llamadas al sistema generan interrupciones para provocar la ejecución de rutinas del *kernel*. Dado que las interrupciones se utilizan de forma tan constante para el procesamiento más crítico en términos de tiempo, se necesita que el tratamiento de las interrupciones sea eficiente para obtener un buen rendimiento del sistema.

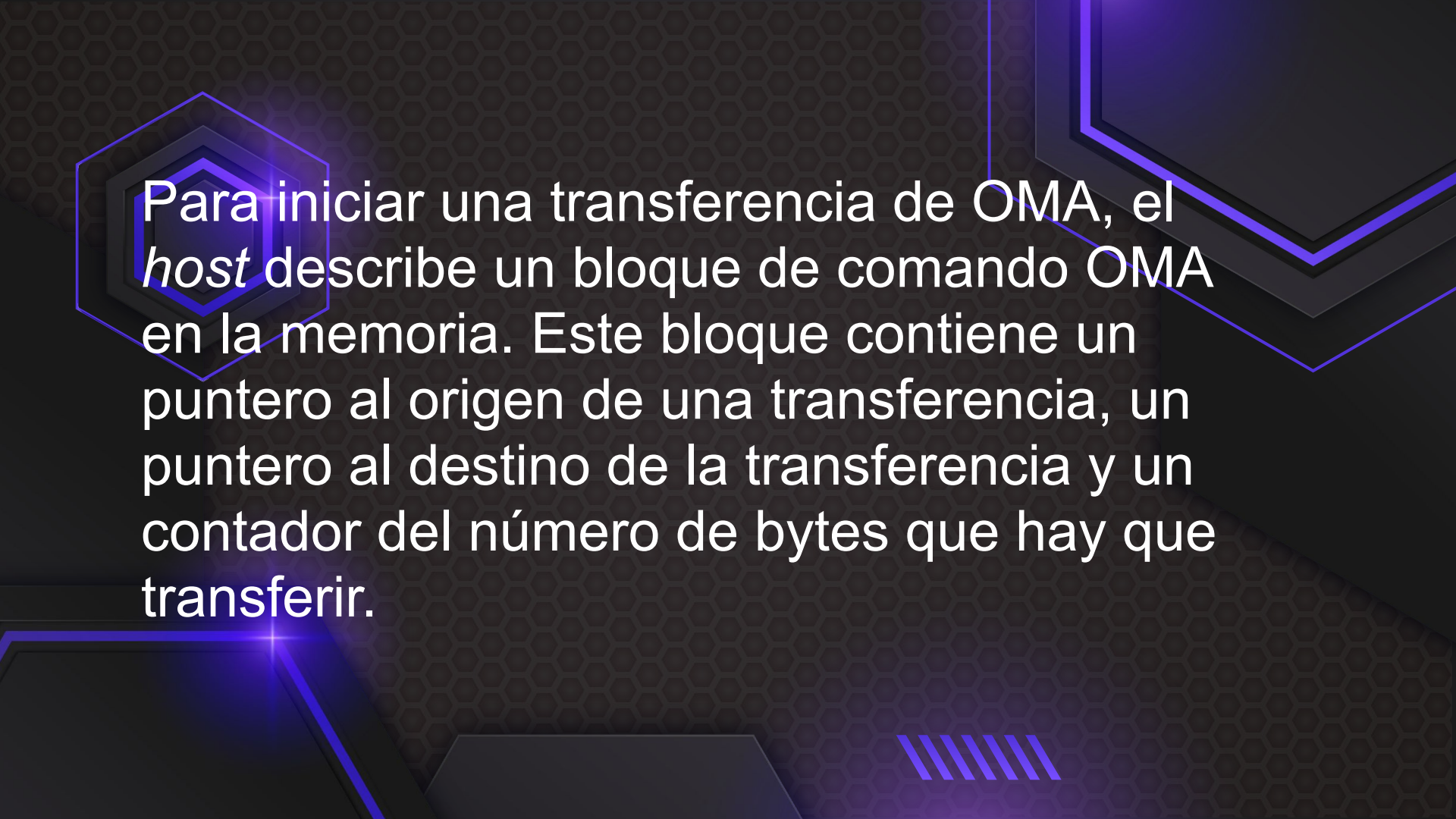


13.2.3


Acceso directo a memoria




Para un dispositivo que realice transferencias de gran tamaño, como por ejemplo una unidad de disco, parece bastante poco apropiado utilizar un caro procesador de propósito general para comprobar dicho estado y para escribir datos en un registro de una controladora de byte en byte, lo cual es un proceso que se denomina **E/S programada**



Para iniciar una transferencia de OMA, el *host* describe un bloque de comando OMA en la memoria. Este bloque contiene un puntero al origen de una transferencia, un puntero al destino de la transferencia y un contador del número de bytes que hay que transferir.

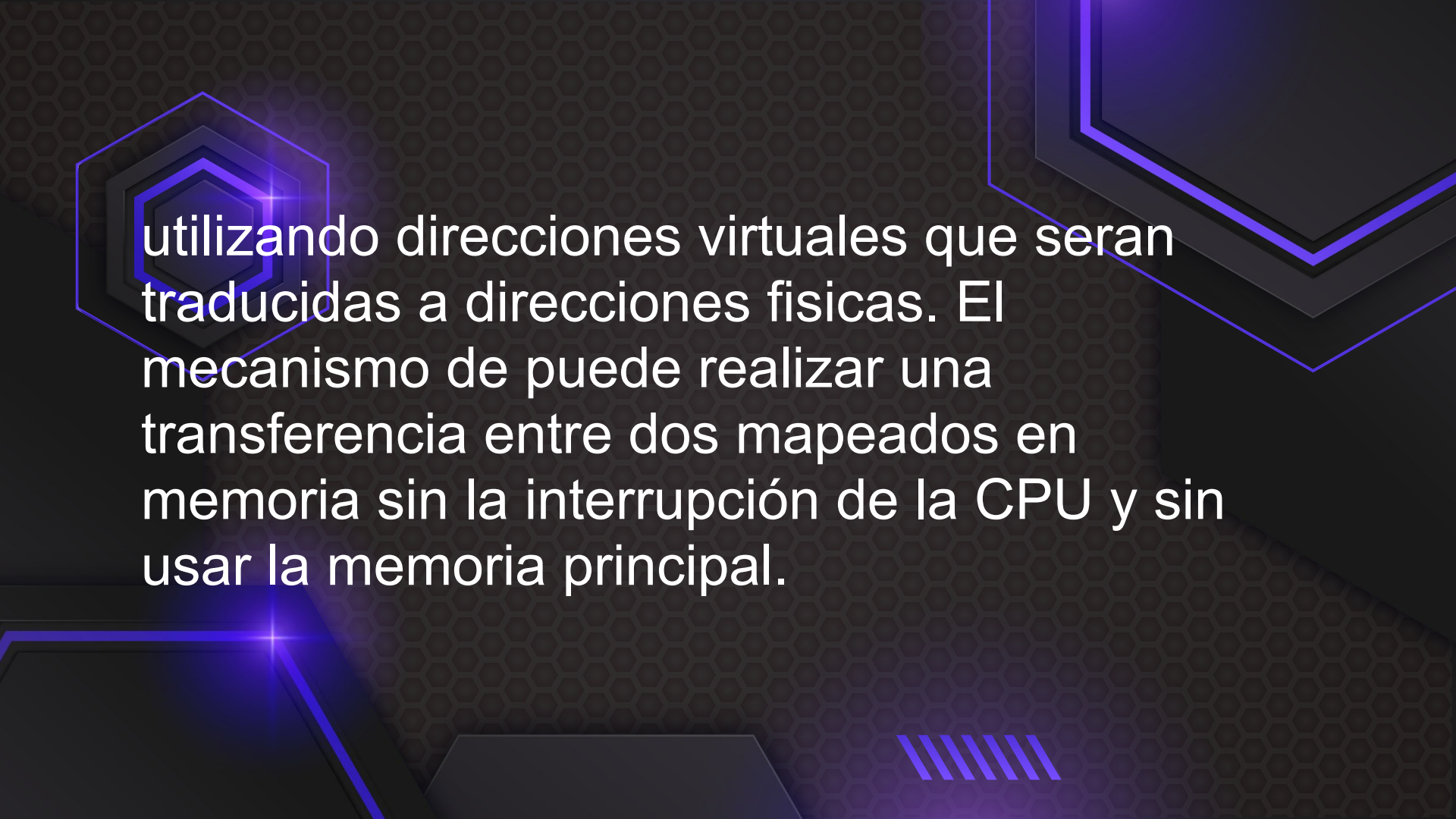




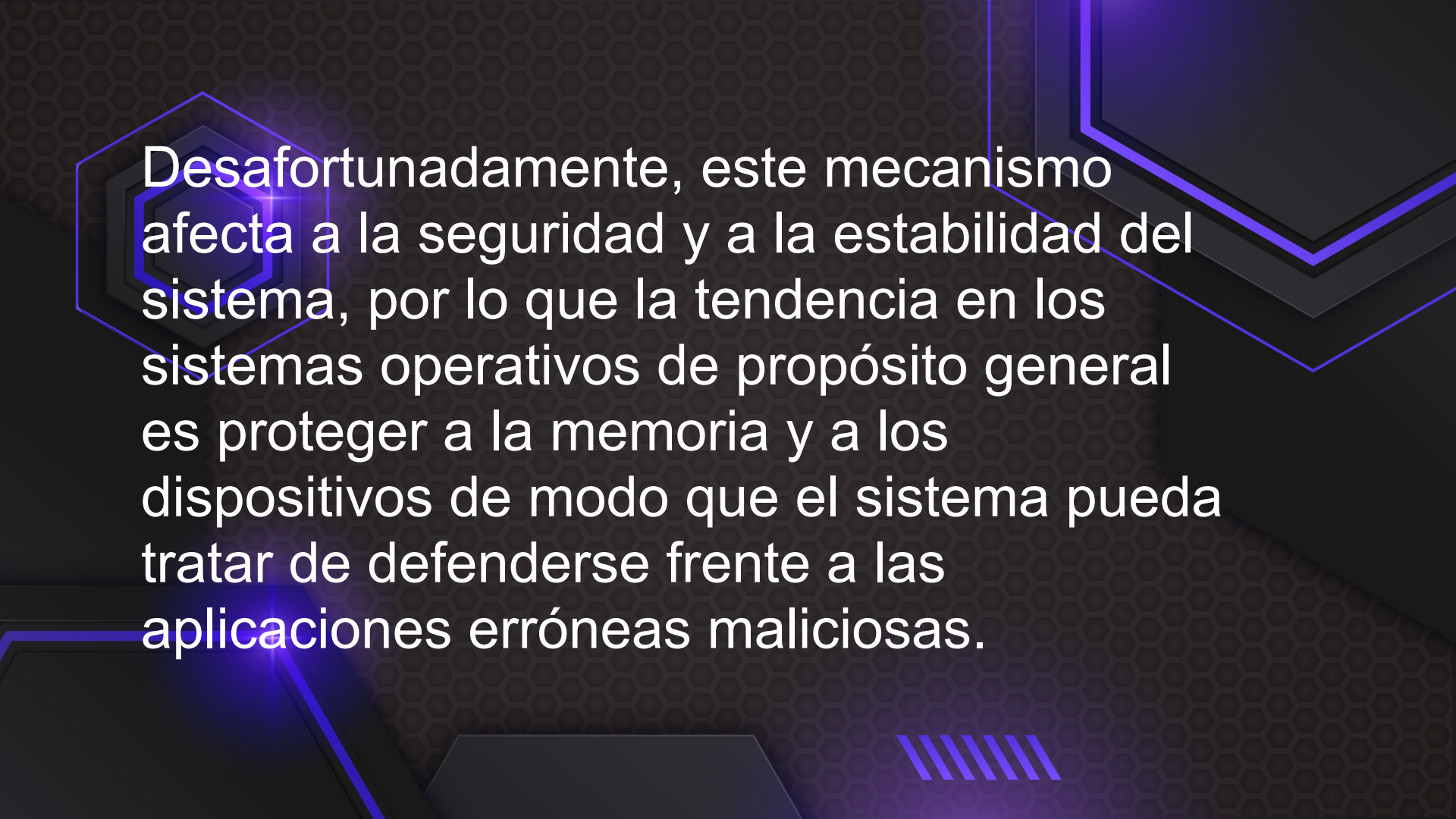
En las computadoras de tipo PC, uno de los componentes estándar es una controladora de OMA simple, y las tarjetas de E/S con **control maestro del bus** para PC suelen contener su propio hardware de OMA de alta velocidad.

Aunque este proceso de **robo de ciclos** puede ralentizar los cálculos realizados por la CPU, descargar el trabajo de transferencia de datos en una controladora de OMA suele mejorar el rendimiento global del sistema. Algunas arquitecturas de computadora utilizan direcciones de memoria física para las operaciones de DMA, mientras que otras realizan un **acceso directo a memoria virtual**





utilizando direcciones virtuales que serán traducidas a direcciones físicas. El mecanismo de puede realizar una transferencia entre dos mapeados en memoria sin la interrupción de la CPU y sin usar la memoria principal.

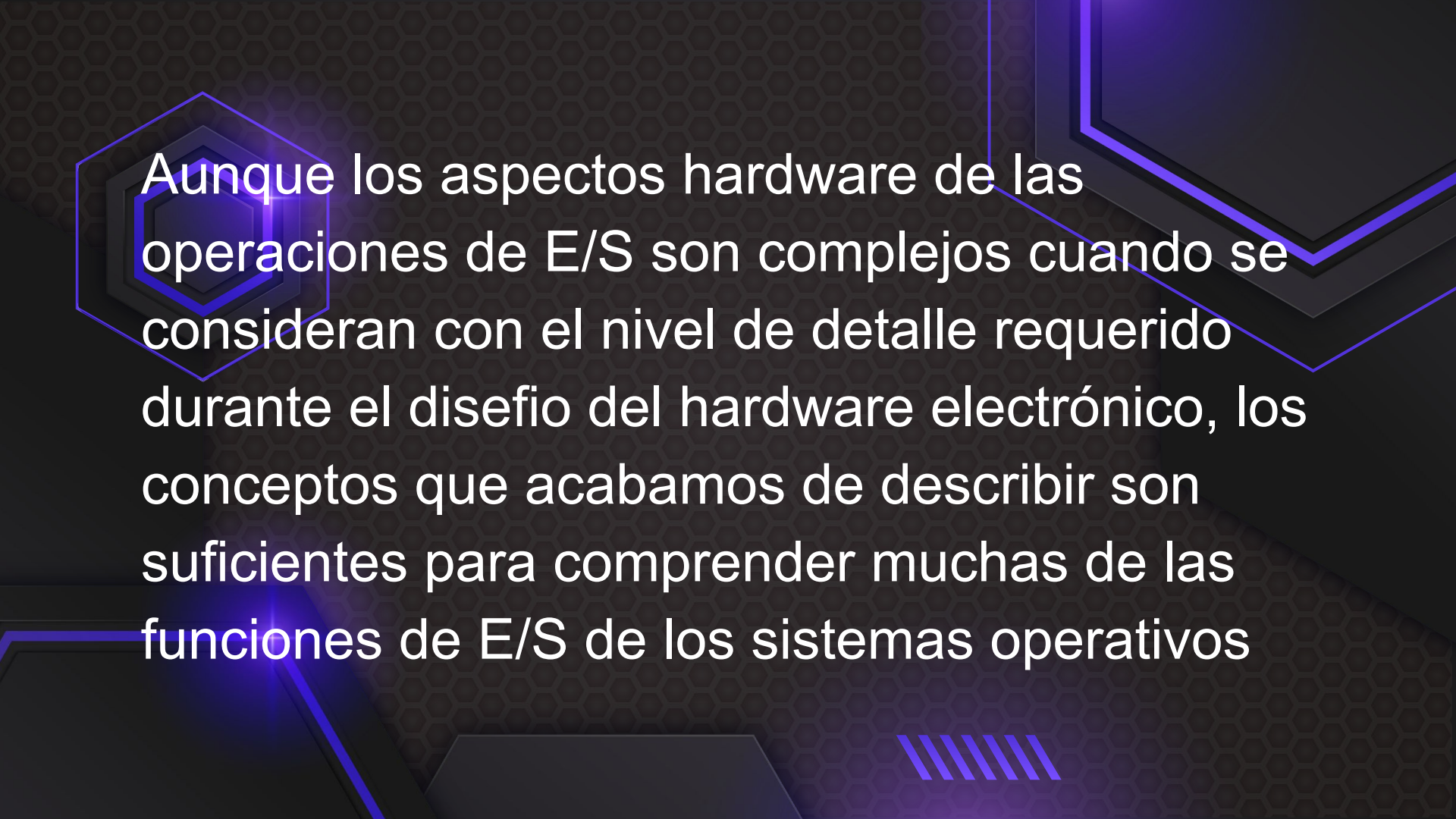


Desafortunadamente, este mecanismo afecta a la seguridad y a la estabilidad del sistema, por lo que la tendencia en los sistemas operativos de propósito general es proteger a la memoria y a los dispositivos de modo que el sistema pueda tratar de defenderse frente a las aplicaciones erróneas maliciosas.


13.2.4

Resumen del hardware E/S





Aunque los aspectos hardware de las operaciones de E/S son complejos cuando se consideran con el nivel de detalle requerido durante el diseño del hardware electrónico, los conceptos que acabamos de describir son suficientes para comprender muchas de las funciones de E/S de los sistemas operativos



4.2 Principios del software de E/S

13.3

Interfaz de E/S de las aplicaciones



El propósito de la capa de controladores de dispositivo es ocultar a ojos del subsistema de E/S del kernel las diferencias existentes entre las controladoras de dispositivo, de forma similar a como las llamadas al sistema de E/S encapsulan el comportamiento de los dispositivos en unas cuantas clases genéricas que ocultan a ojos de las aplicaciones las diferencias hardware.





Flujo de caracteres o bloque

Un dispositivo de flujo de caracteres transfiere los bytes uno a uno, mientras que un dispositivo de bloque transfiere un bloque de bytes como una sola unidad.

Acceso secuencial o aleatorio

Un dispositivo secuencial transfiere los datos en un orden fijo determinado por el dispositivo, mientras que el usuario de un dispositivo de acceso aleatorio puede instruir al dispositivo para que se posicione en cualquiera de las ubicaciones disponibles de almacenamiento de datos.

Síncrono o asíncrono

Un dispositivo síncrono realiza transferencias de datos con tiempos de respuesta predecibles. Un dispositivo asíncrono exhibe unos tiempos de respuesta irregulares o no predecibles.





Compatible o dedicado

Un dispositivo compatible puede ser usado de forma concurrente por varios procesos o hebras; un dispositivo dedicado no puede ser compartido de esta forma.

Velocidad de operación

Las velocidades de los dispositivos van desde unos pocos bytes por segundo a unos cuantos gigabytes por segundo.

Lectura-escritura, solo lectura o solo escritura

Algunos dispositivos realizan tanto entrada como salida, pero otros sólo soportan una única dirección de transferencia de los datos.





Sistemas de acceso

- E/S de bloque
- E/S de flujo de caracteres
- Acceso a archivos mapeados en memoria
- Sockets de red



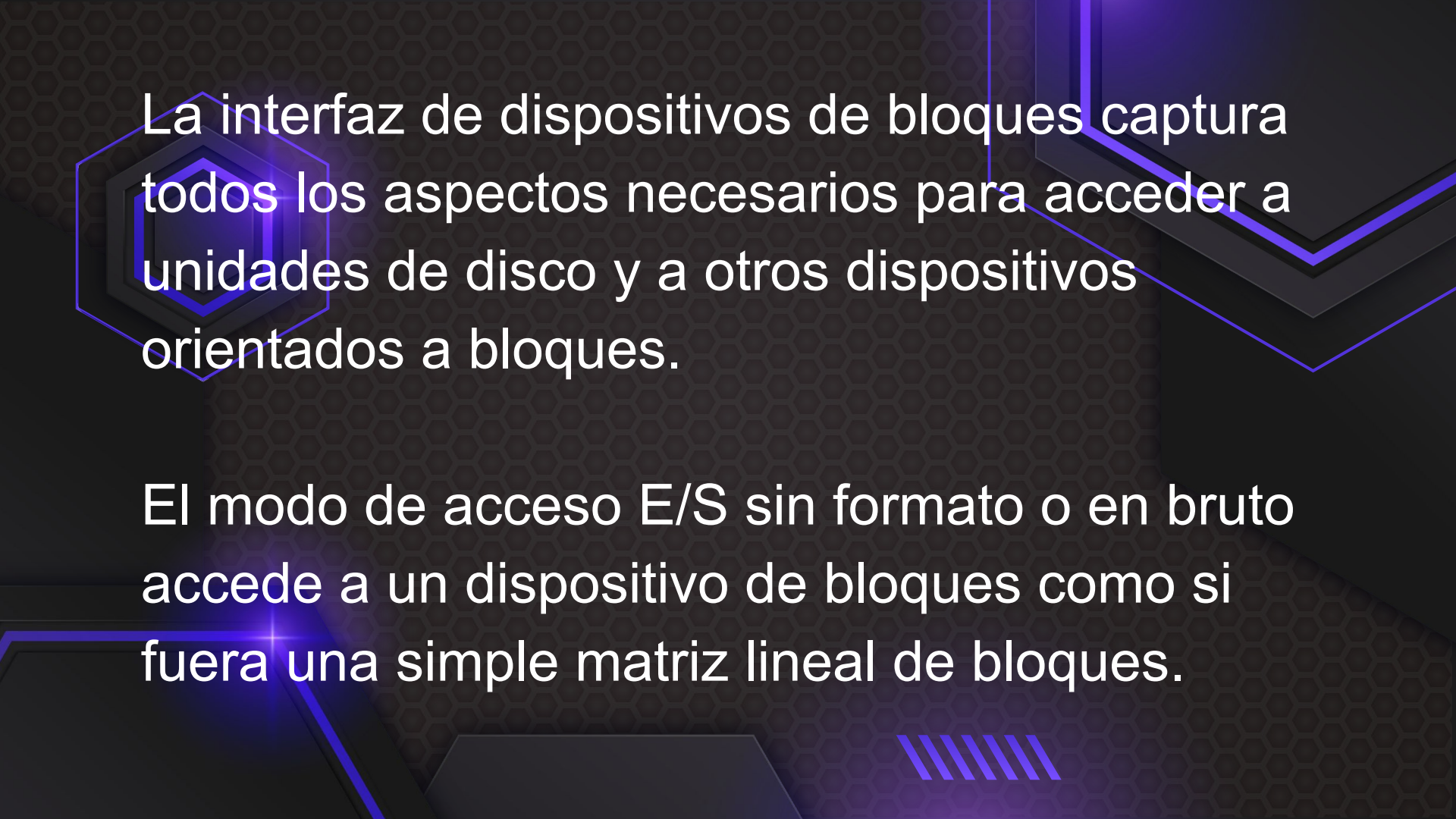
Sistemas operativos

- Tienen un escape o puerta trasera que pasa de manera transparente una serie de comandos arbitrarios desde una aplicación a un controlador de dispositivo.

13.3.1


**Dispositivos de
bloques y de
caracteres**

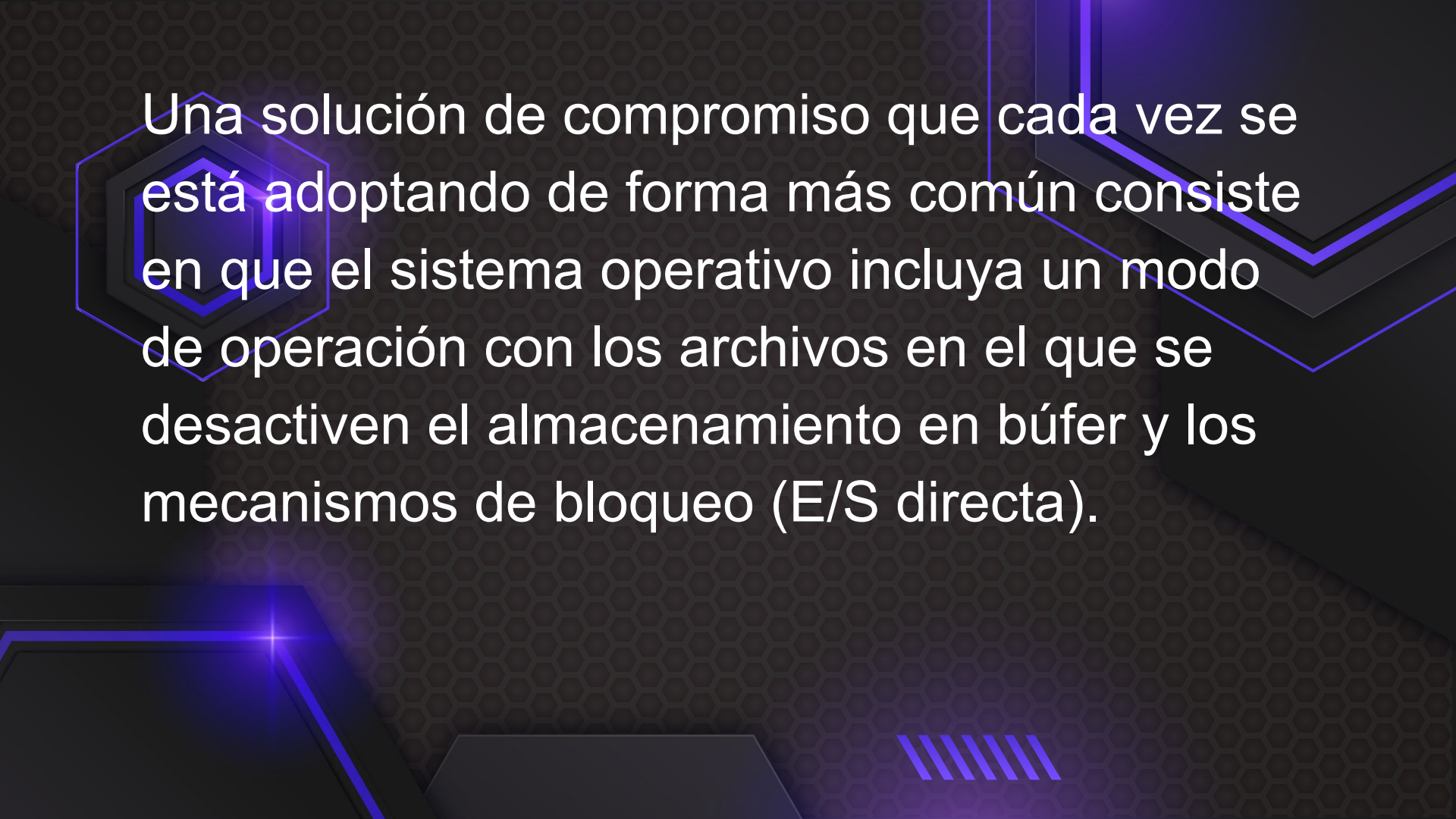




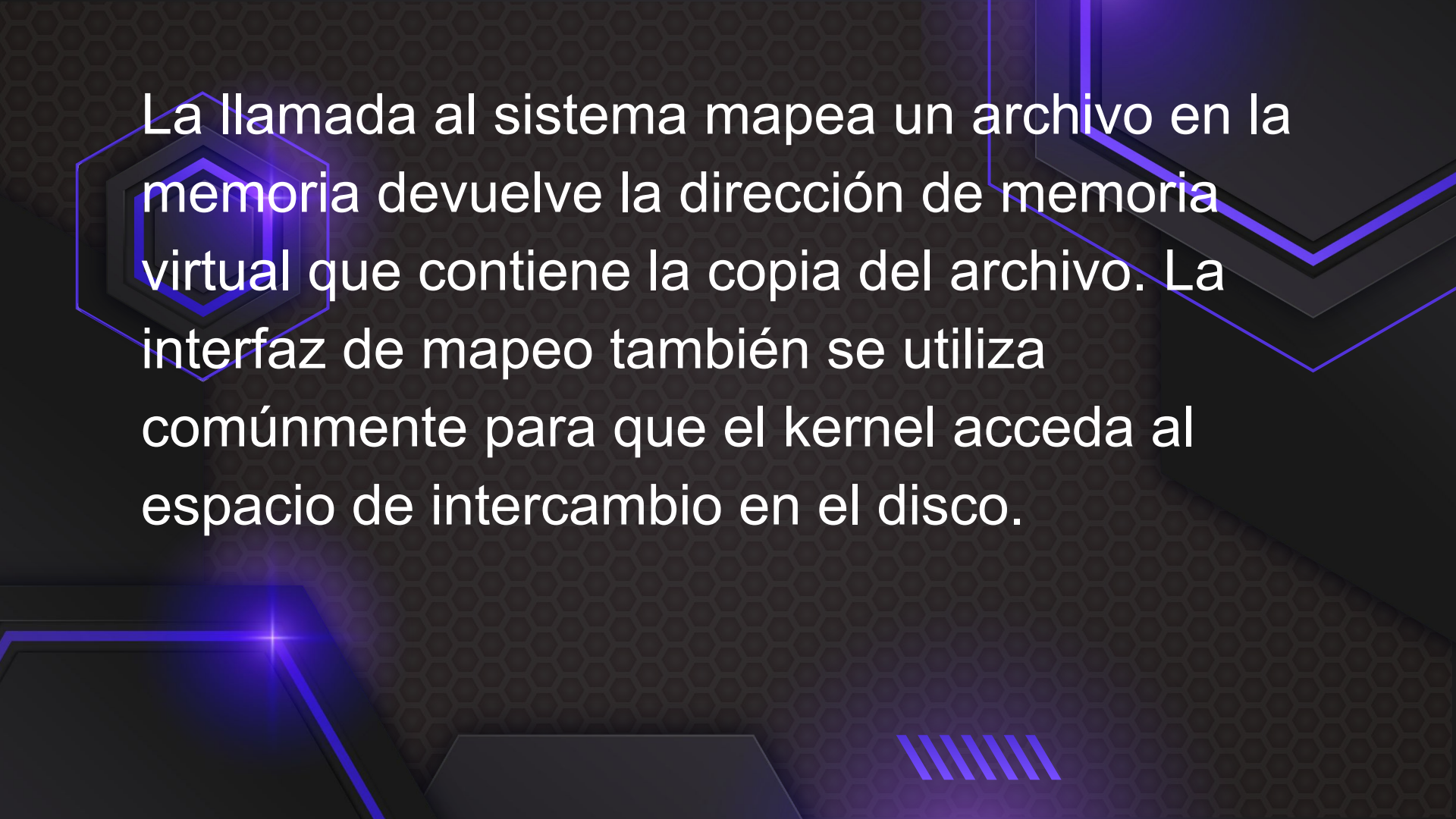
La interfaz de dispositivos de bloques captura todos los aspectos necesarios para acceder a unidades de disco y a otros dispositivos orientados a bloques.

El modo de acceso E/S sin formato o en bruto accede a un dispositivo de bloques como si fuera una simple matriz lineal de bloques.





Una solución de compromiso que cada vez se está adoptando de forma más común consiste en que el sistema operativo incluya un modo de operación con los archivos en el que se desactiven el almacenamiento en búfer y los mecanismos de bloqueo (E/S directa).




La llamada al sistema mapea un archivo en la memoria devuelve la dirección de memoria virtual que contiene la copia del archivo. La interfaz de mapeo también se utiliza comúnmente para que el kernel acceda al espacio de intercambio en el disco.

13.3.2


Dispositivos de red

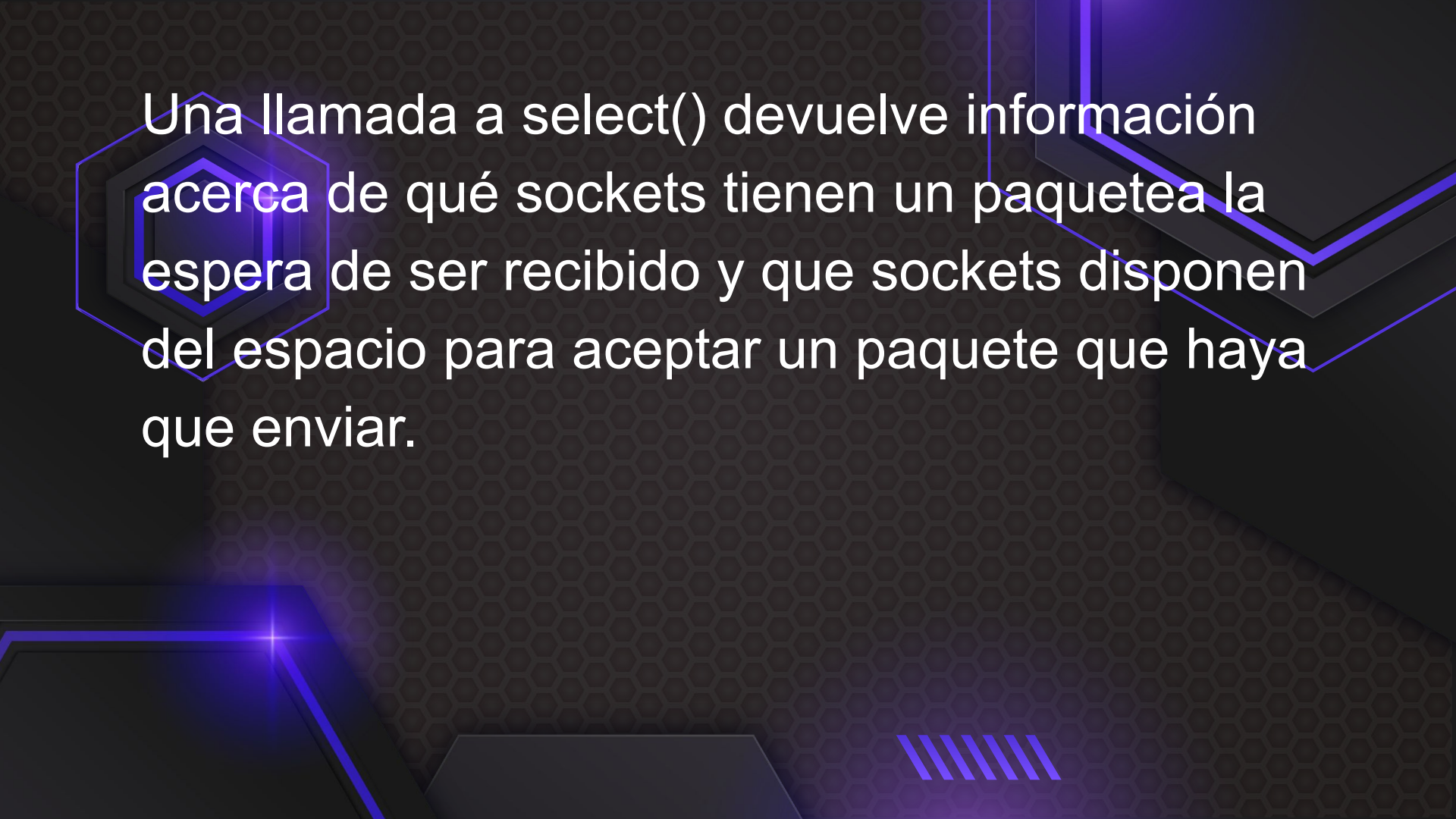


A decorative graphic consisting of several concentric hexagons. The innermost hexagon is a solid dark blue, and the outer ones are outlines in a lighter blue/purple color, creating a layered effect.

Los sockets son aquellos con los que podemos conectar una aplicación con otra.

Para soportar la implementación de servidores, la interfaz de sockets también proporciona una función denominada `select()` que gestiona un conjunto de sockets.

A decorative graphic consisting of several parallel diagonal lines in a light blue/purple color, slanted upwards from left to right.



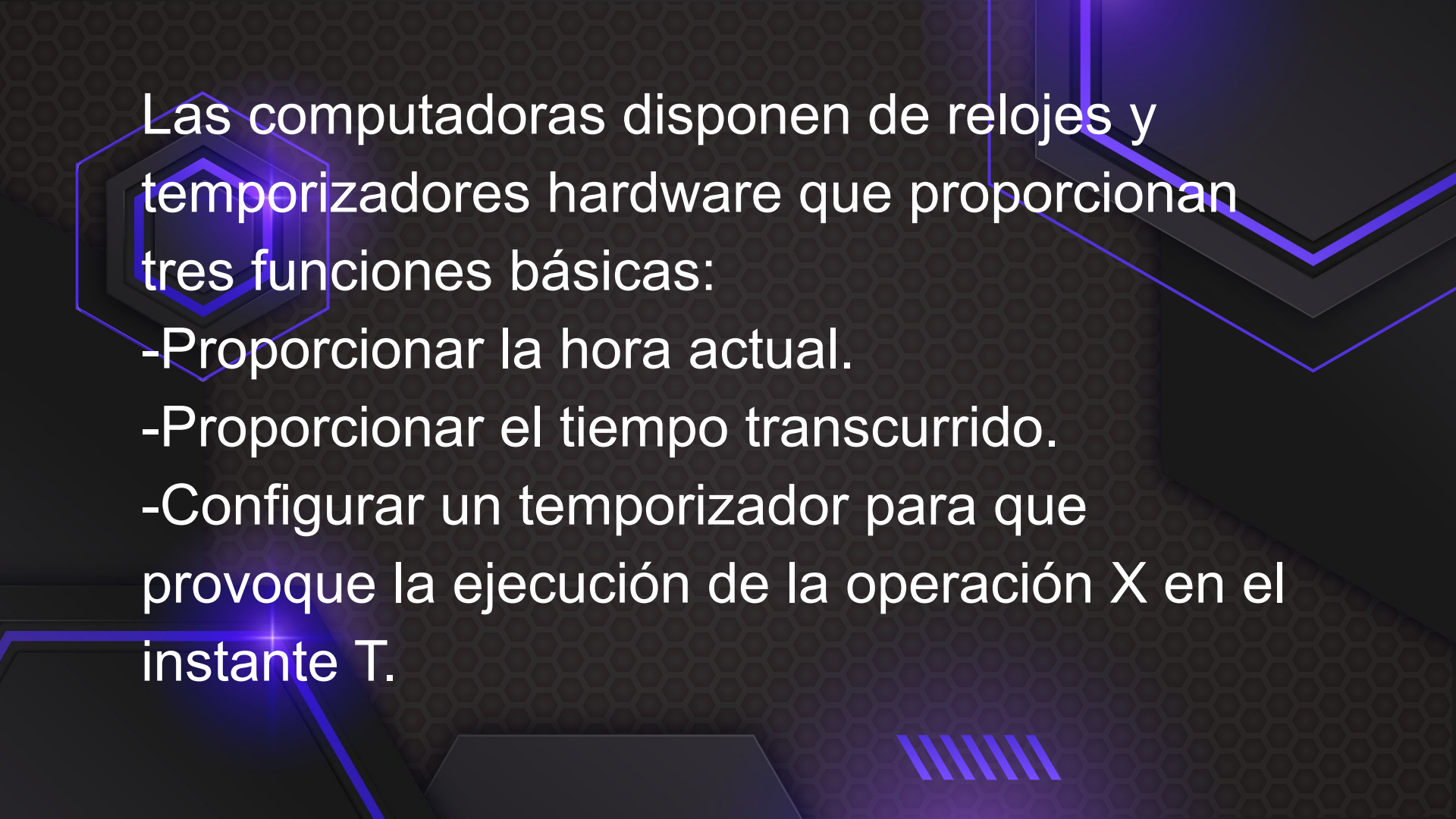
Una llamada a `select()` devuelve información acerca de qué sockets tienen un paquete a la espera de ser recibido y que sockets disponen del espacio para aceptar un paquete que haya que enviar.




13.3.2

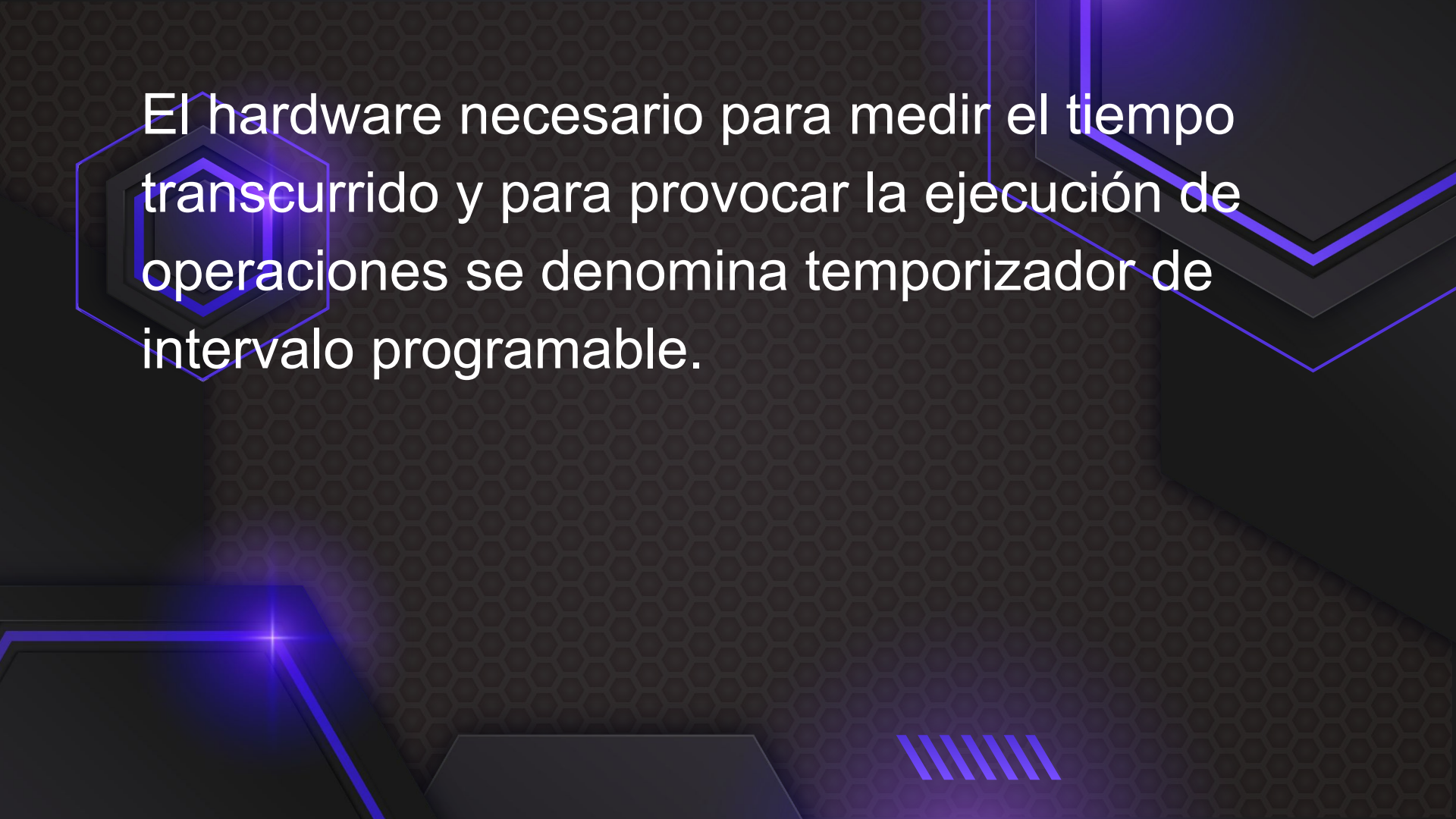
Dispositivos de red





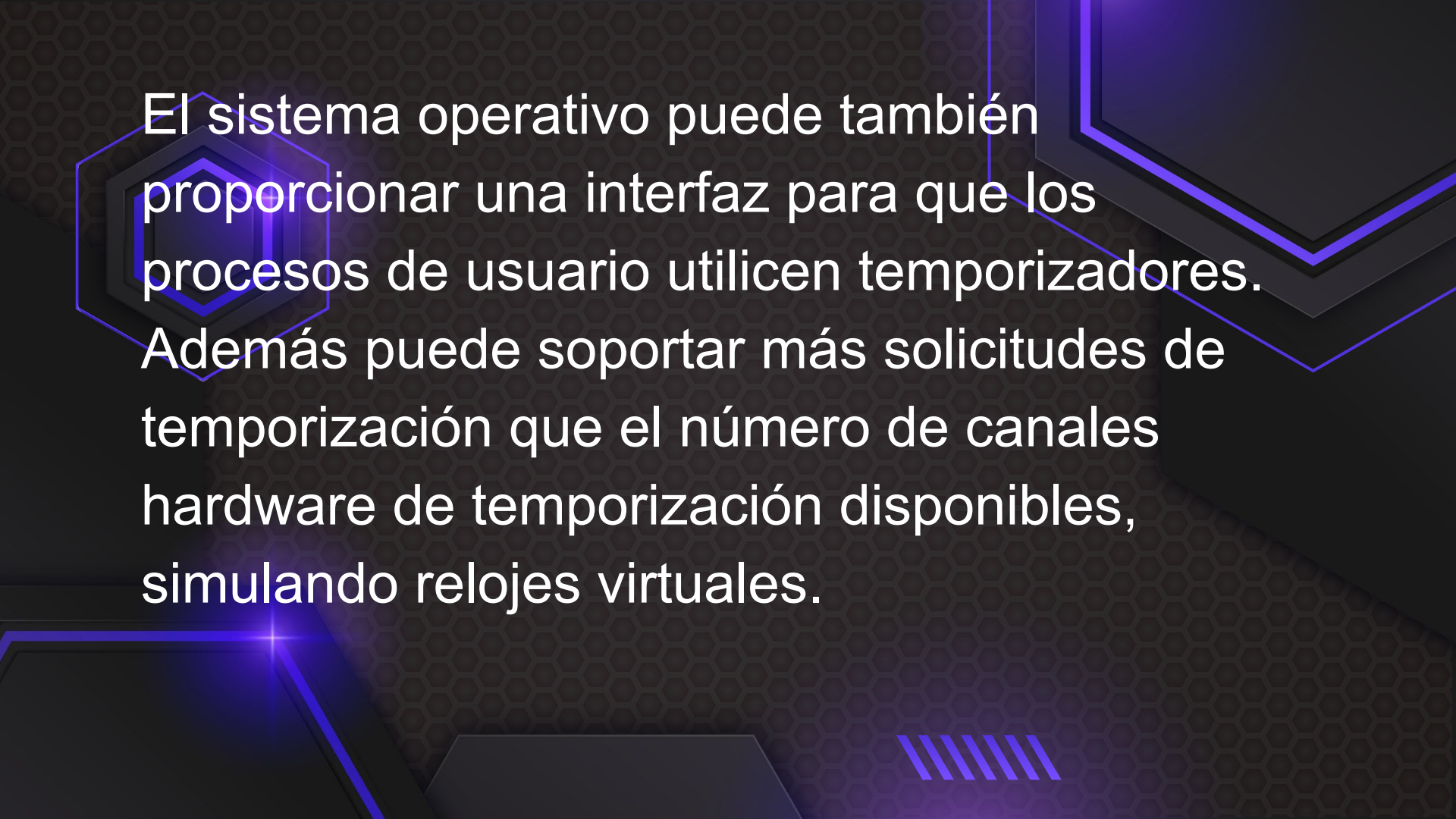
Las computadoras disponen de relojes y temporizadores hardware que proporcionan tres funciones básicas:

- Proporcionar la hora actual.
 - Proporcionar el tiempo transcurrido.
 - Configurar un temporizador para que provoque la ejecución de la operación X en el instante T.
- 




El hardware necesario para medir el tiempo transcurrido y para provocar la ejecución de operaciones se denomina temporizador de intervalo programable.






El sistema operativo puede también proporcionar una interfaz para que los procesos de usuario utilicen temporizadores. Además puede soportar más solicitudes de temporización que el número de canales hardware de temporización disponibles, simulando relojes virtuales.

A decorative graphic consisting of several concentric hexagons in shades of blue and purple, located on the left side of the slide.

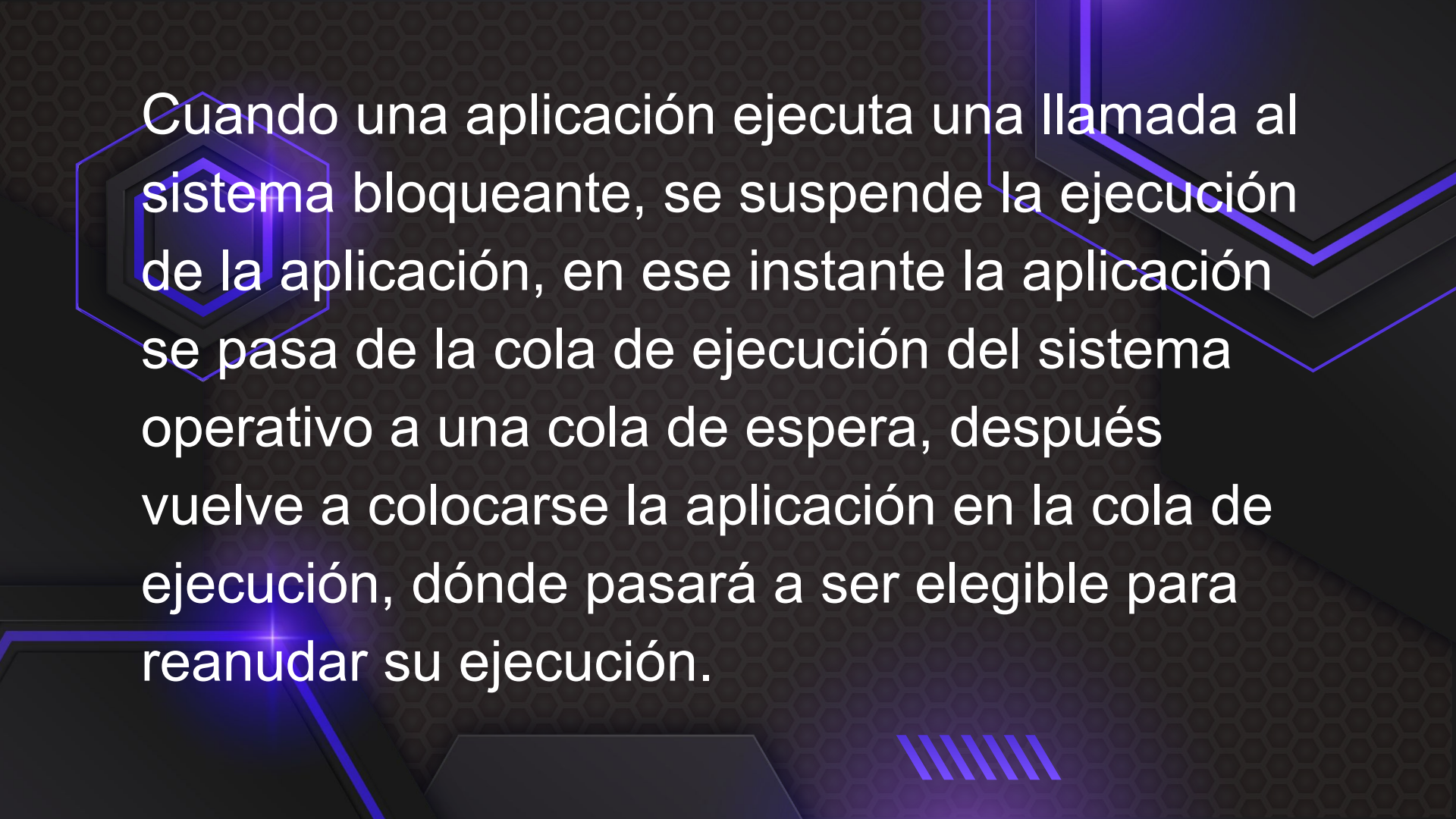
El kernel mantiene una lista de interrupciones
deseadas por sus rutinas y por las solicitudes.
El kernel considera el temporizador para dar
servicio a la interrupción más próxima.

Después se señala dicho suceso al proceso
solicitante y recarga el temporizador con la
siguiente interrupción de la secuencia.

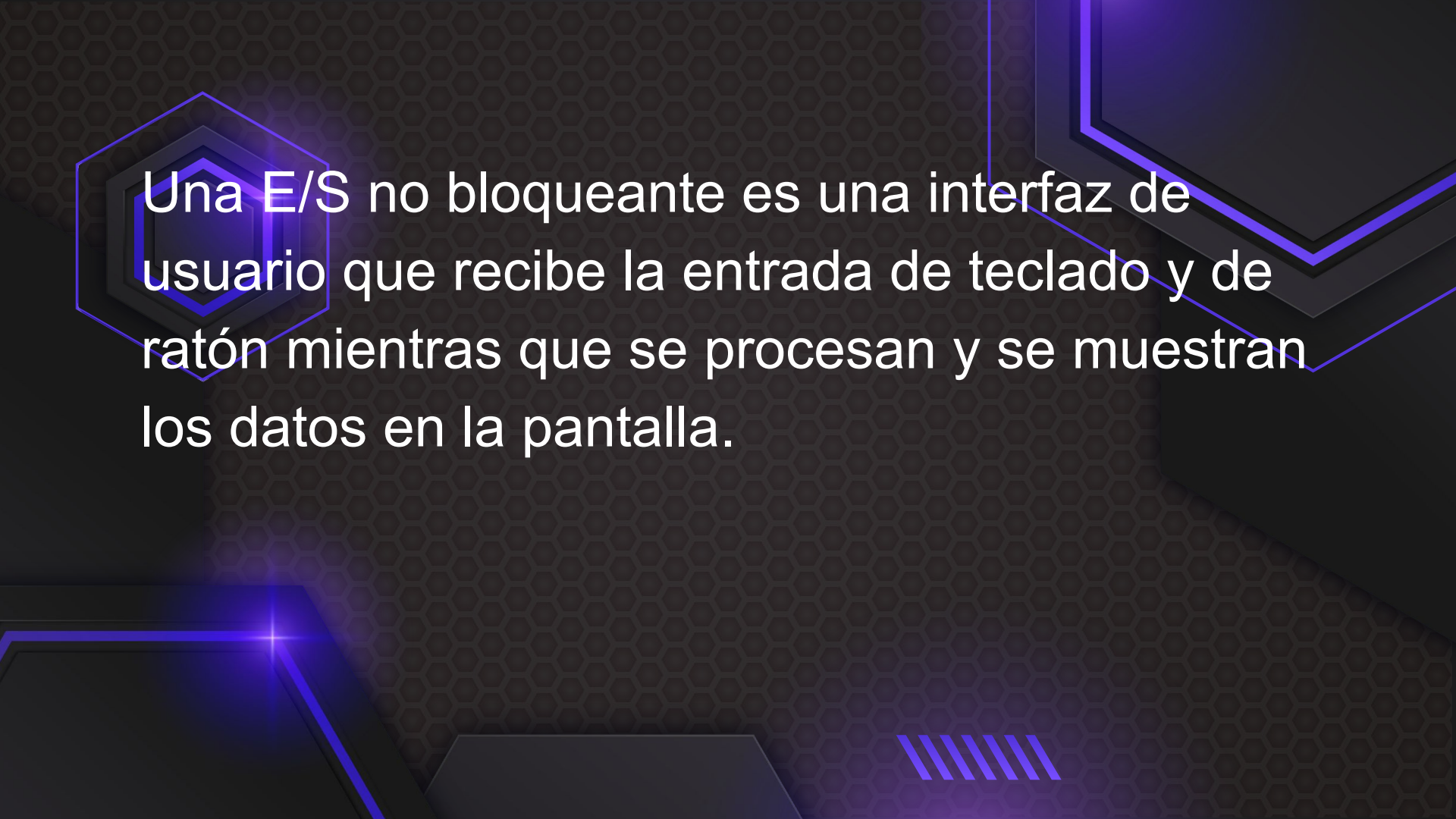
A decorative graphic consisting of several parallel diagonal lines in shades of blue and purple, located in the bottom right corner of the slide.

13.3.4 E/S bloqueante y no bloqueante





Cuando una aplicación ejecuta una llamada al sistema bloqueante, se suspende la ejecución de la aplicación, en ese instante la aplicación se pasa de la cola de ejecución del sistema operativo a una cola de espera, después vuelve a colocarse la aplicación en la cola de ejecución, dónde pasará a ser elegible para reanudar su ejecución.



Una E/S no bloqueante es una interfaz de usuario que recibe la entrada de teclado y de ratón mientras que se procesan y se muestran los datos en la pantalla.

4.2 Principios del software de E/S

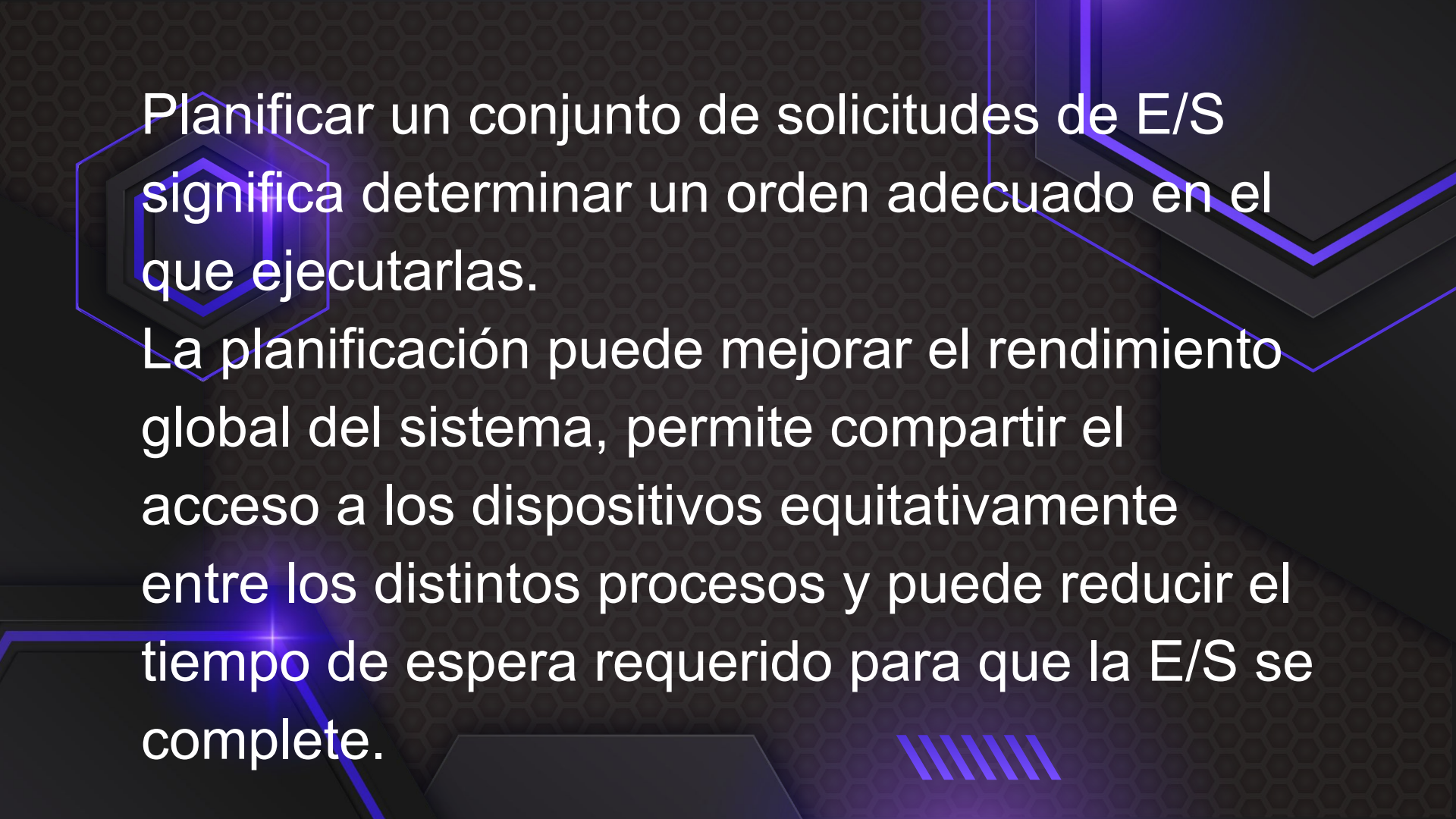
13.4 Subsistema de E/S del kernel



13.4.1


Planificación de E/S

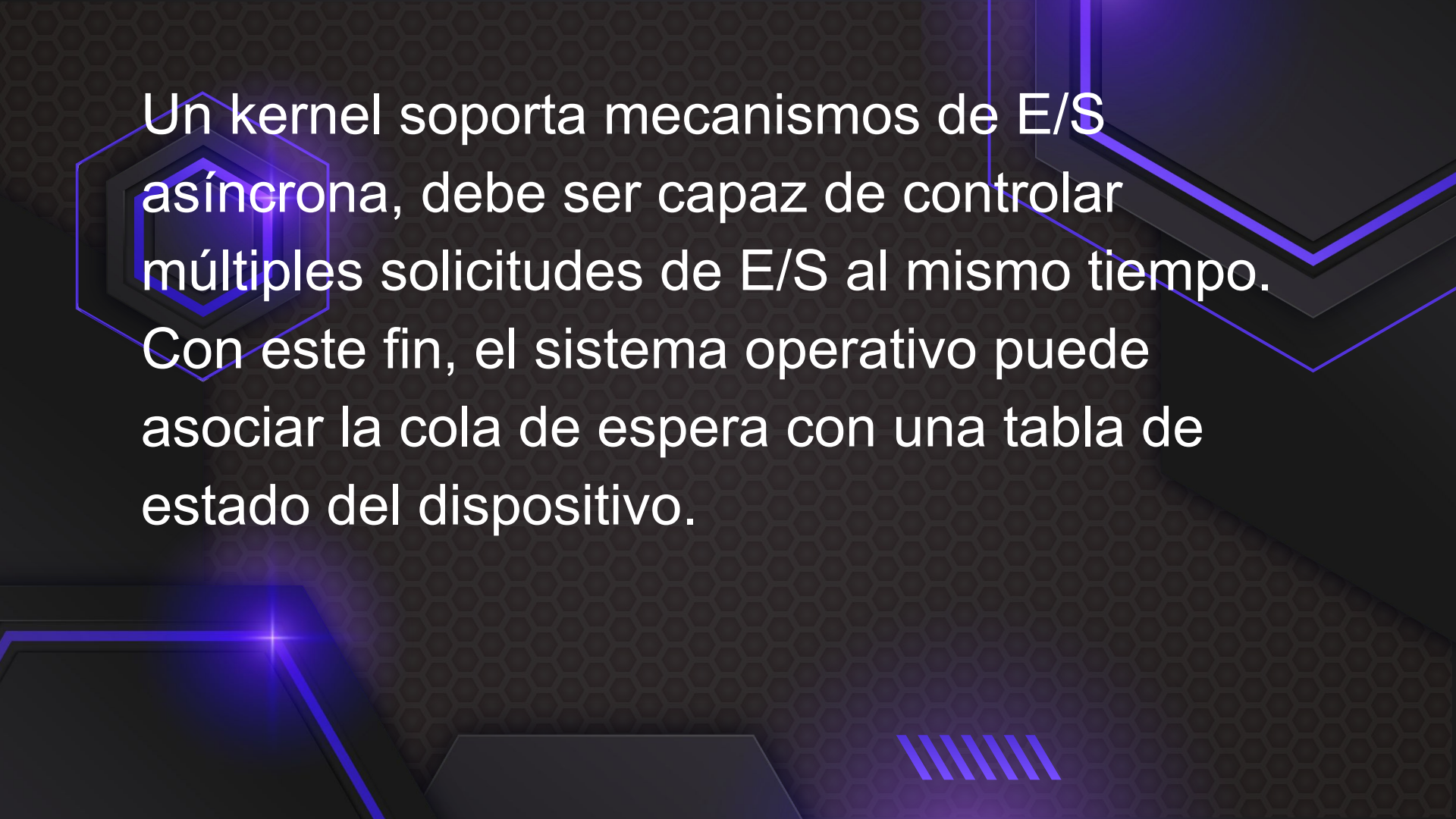




Planificar un conjunto de solicitudes de E/S significa determinar un orden adecuado en el que ejecutarlas.

La planificación puede mejorar el rendimiento global del sistema, permite compartir el acceso a los dispositivos equitativamente entre los distintos procesos y puede reducir el tiempo de espera requerido para que la E/S se complete.





Un kernel soporta mecanismos de E/S asíncrona, debe ser capaz de controlar múltiples solicitudes de E/S al mismo tiempo. Con este fin, el sistema operativo puede asociar la cola de espera con una tabla de estado del dispositivo.

13.4.2

Almacenamiento de búfer



Un búfer es un área de memoria que almacena datos mientras se están transfiriendo entre dos dispositivos o entre un dispositivo y una aplicación.



Se realiza por 3 razones

01

Realizar una adaptación de velocidades entre el productor y el consumidor de un flujo de datos.

02

Realizar la adaptación entre dispositivos que tengan diferentes tamaños de transferencia de datos.

03

Soportar la semántica de copia en la E/S de las aplicaciones.



La copia de datos entre búferes del kernel y el espacio de datos de la aplicación resulta bastante común en los sistemas operativos.

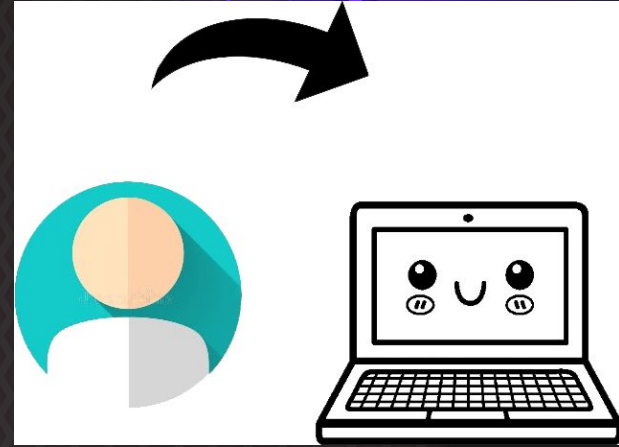


13.4.3

Almacenamiento de caché

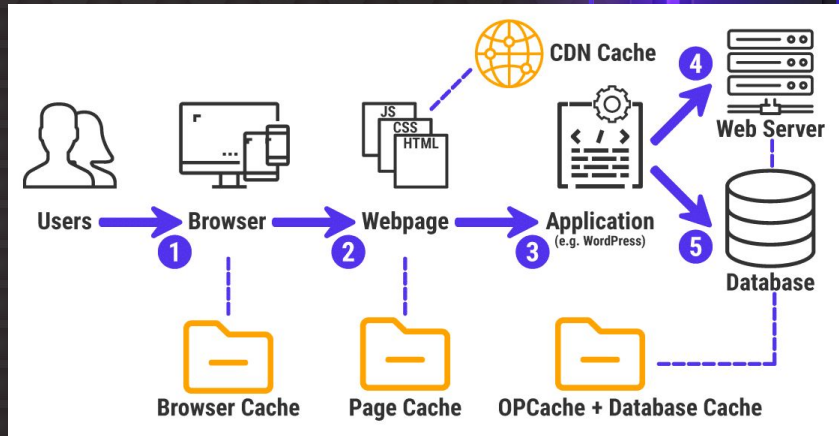


Es una región de
memoria rápida
que alberga copias
de ciertos datos.

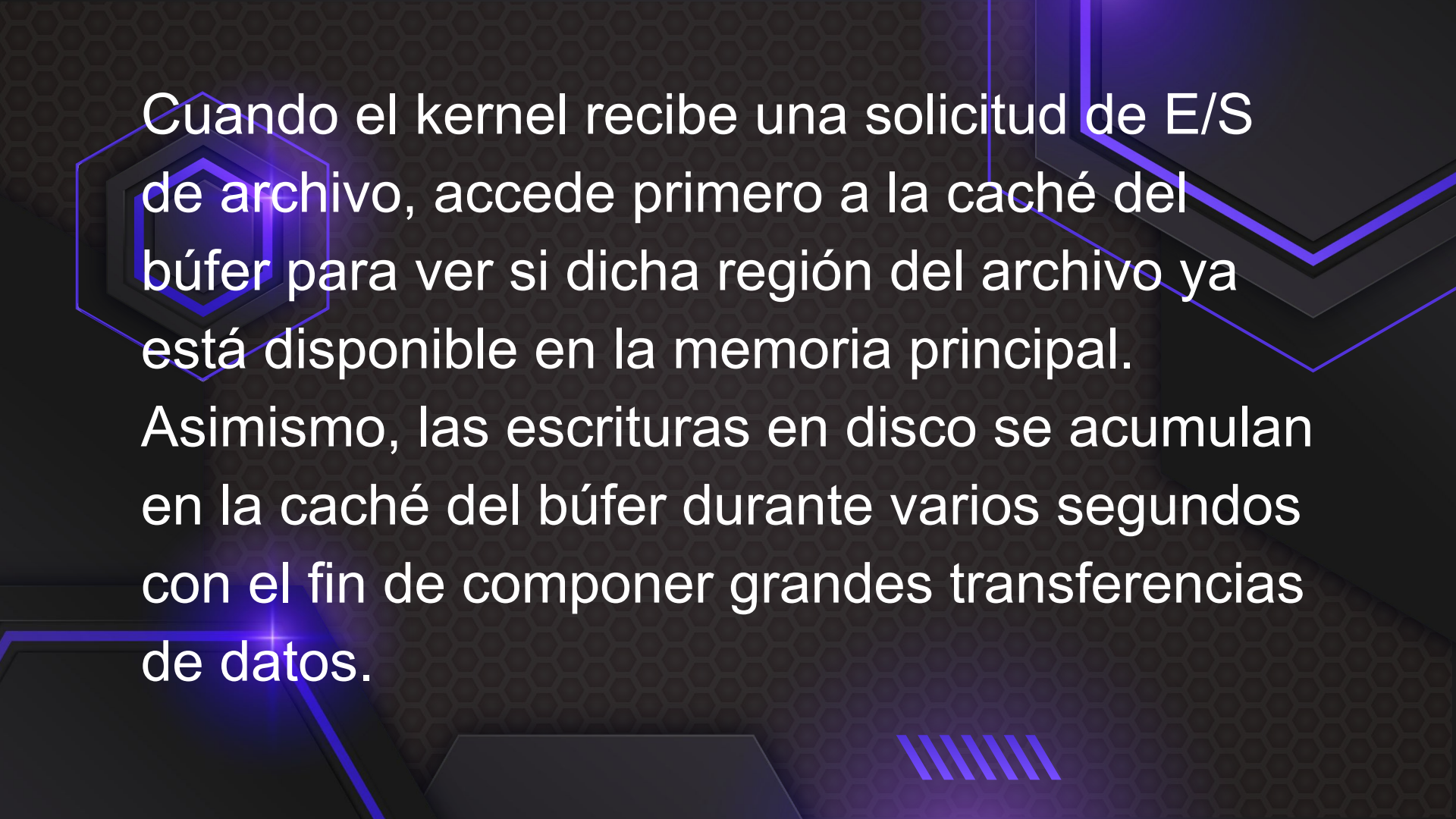


Diferencia entre búfer y cache

Un búfer puede almacenar la única copia existente de un elemento de datos.




Un caché es un dispositivo de almacenamiento más rápido



Cuando el kernel recibe una solicitud de E/S de archivo, accede primero a la caché del búfer para ver si dicha región del archivo ya está disponible en la memoria principal.

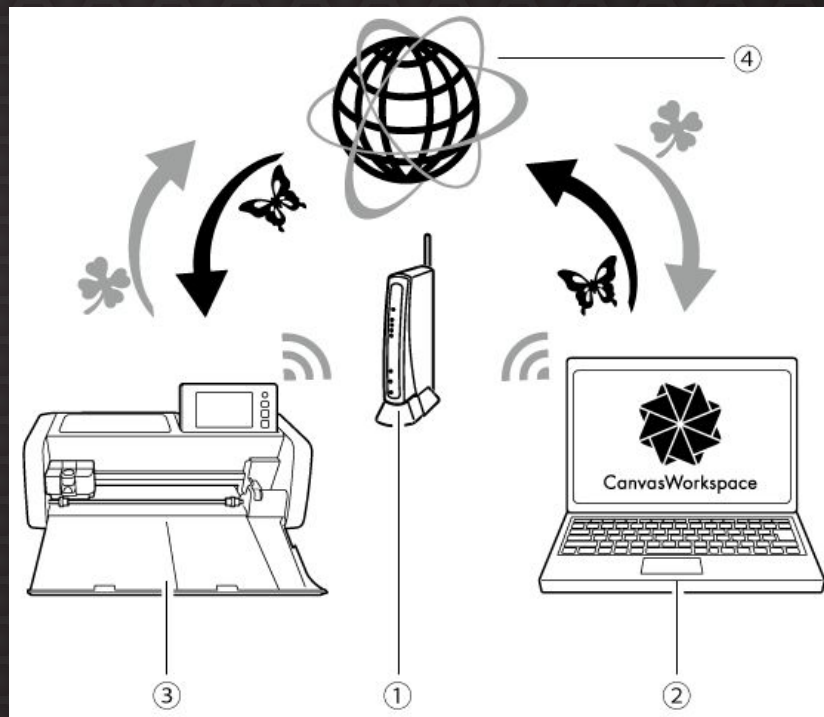
Asimismo, las escrituras en disco se acumulan en la caché del búfer durante varios segundos con el fin de componer grandes transferencias de datos.

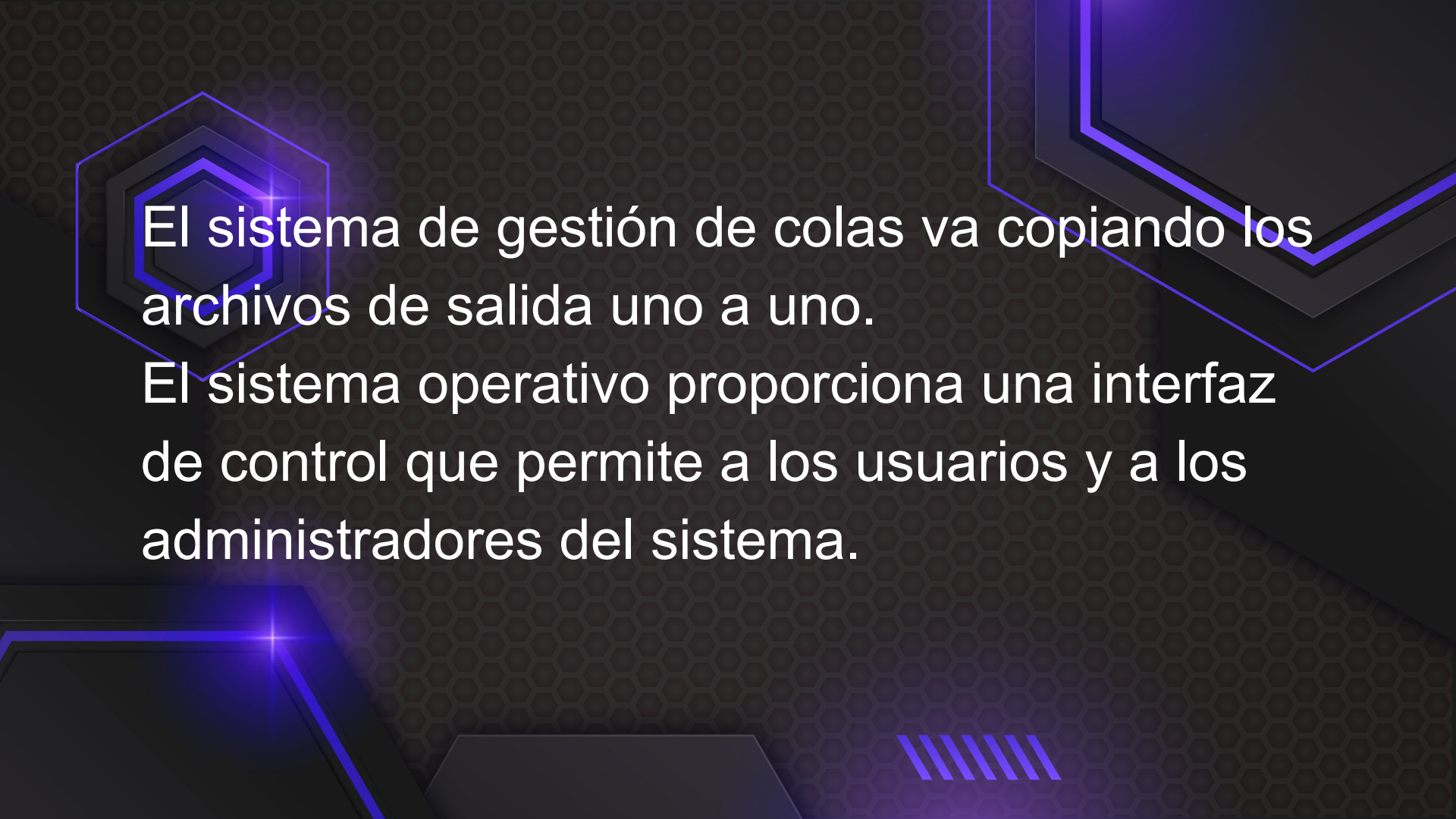


13.4.4 Gestión de colas y reserva de dispositivos.



Una cola de dispositivo es un búfer que almacena la salida dirigida a un dispositivo.






El sistema de gestión de colas va copiando los archivos de salida uno a uno.

El sistema operativo proporciona una interfaz de control que permite a los usuarios y a los administradores del sistema.

13.4.5


Tratamiento de errores.




A decorative graphic consisting of several concentric hexagons with a blue-to-purple gradient, located to the left of the first text block.

Un sistema operativo que utilice memoria protegida puede defenderse frente a muchos tipos de errores de hardware y de las aplicaciones.

Los dispositivos y las transferencias de E/S pueden fallar de muchas formas debido a razones transitorias.

A decorative graphic consisting of several parallel diagonal lines with a blue-to-purple gradient, located in the bottom right corner of the slide.

A decorative graphic consisting of several concentric hexagons. The innermost hexagon is a solid blue color, while the others are outlines in a lighter blue shade. They are arranged in a slightly offset, layered fashion.

Una llamada de E/S al sistema devolverá un bit de información acerca a estado de la llamada, mediante el que se indicará si ésta ha tenido éxito o no.


Ciertos tipos de hardware pueden proporcionar información de error altamente detallada.

A decorative graphic consisting of several parallel diagonal lines. The lines are a solid blue color and are arranged in a slightly offset, layered fashion.



Los errores están relacionados con las cuestiones de protección.

Un proceso de usuario puede intentar accidental o interrumpir la operación normal de un sistema, tratando de ejecutar instrucciones de E/S ilegales.



Interruptor
software al
monitor



leer

.
. .
Llamada al
sistema n

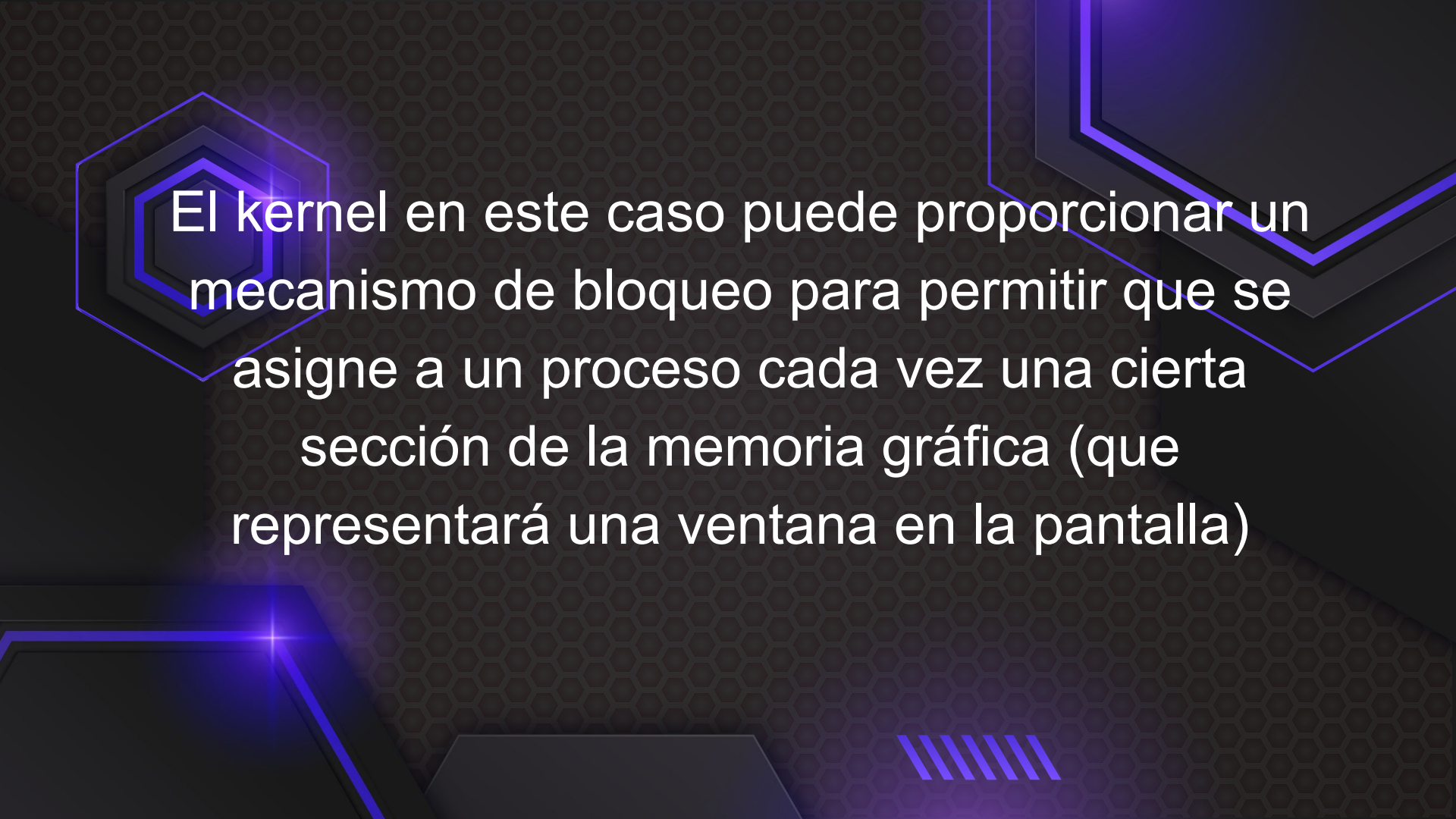
.
. .
.

Kernel

Realizar
E/S

Volver al
usuario

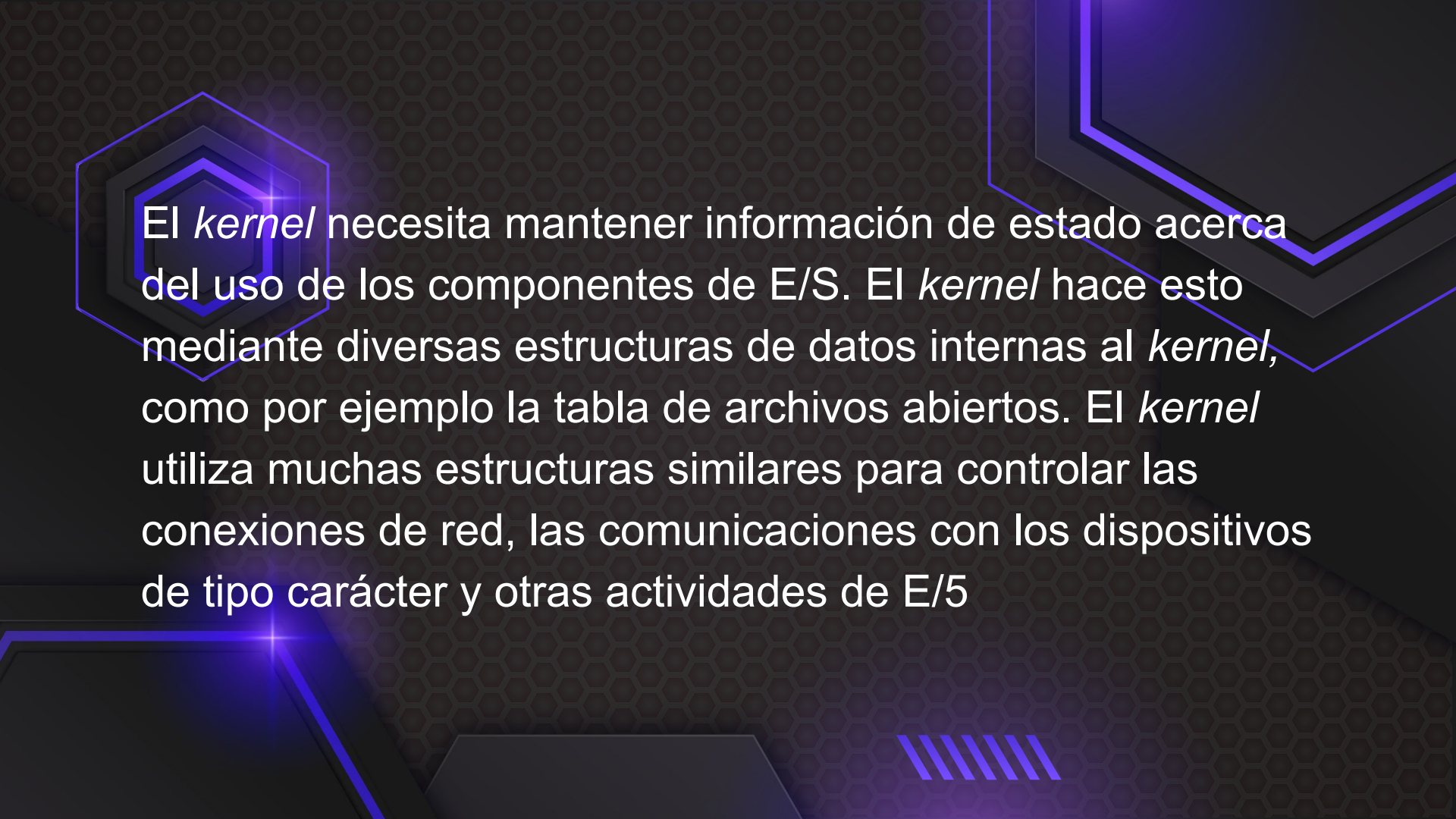
Programa
de usuario




El kernel en este caso puede proporcionar un mecanismo de bloqueo para permitir que se asigne a un proceso cada vez una cierta sección de la memoria gráfica (que representará una ventana en la pantalla)

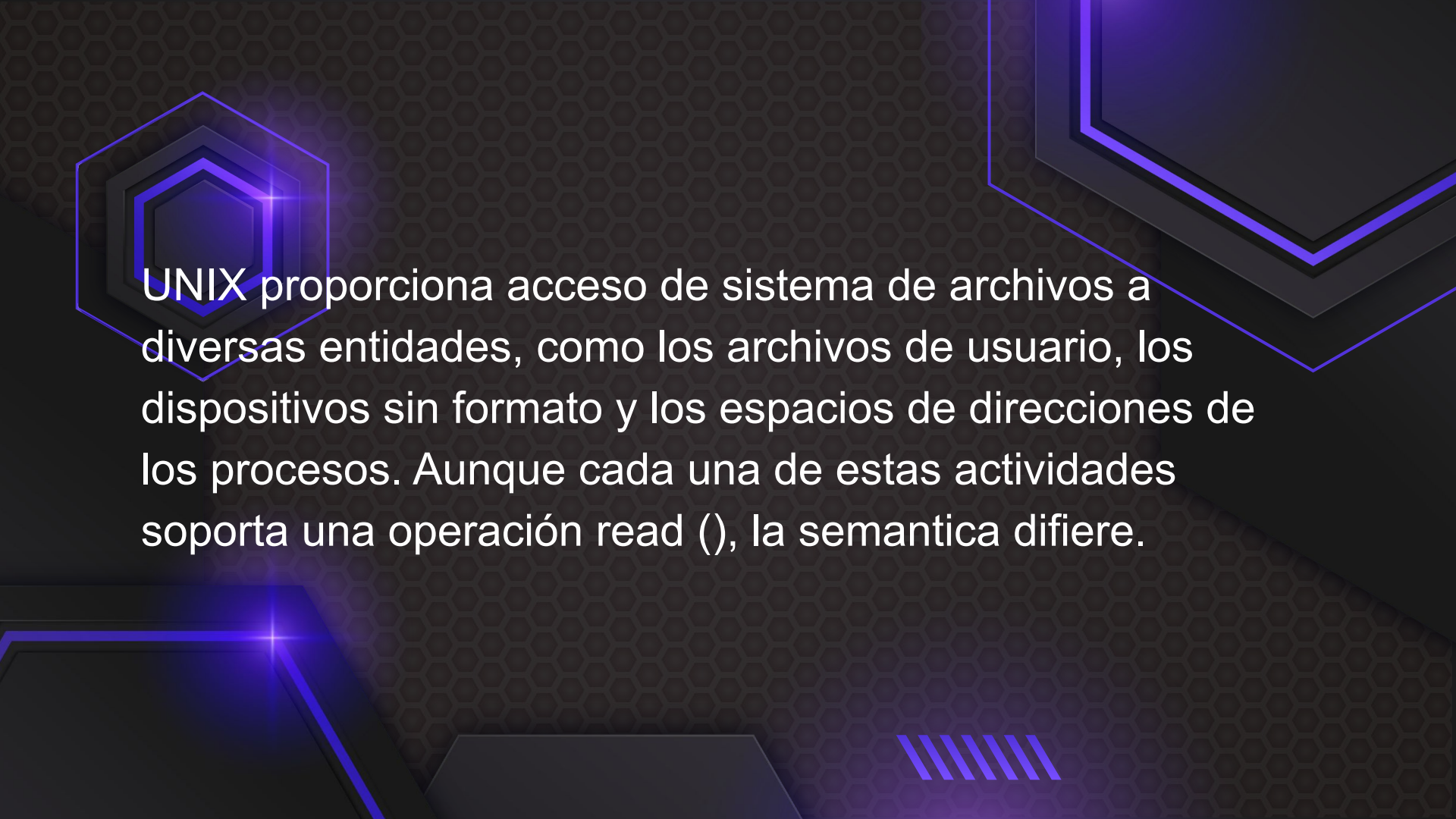
13.4.7 Estructuras de datos del kernel






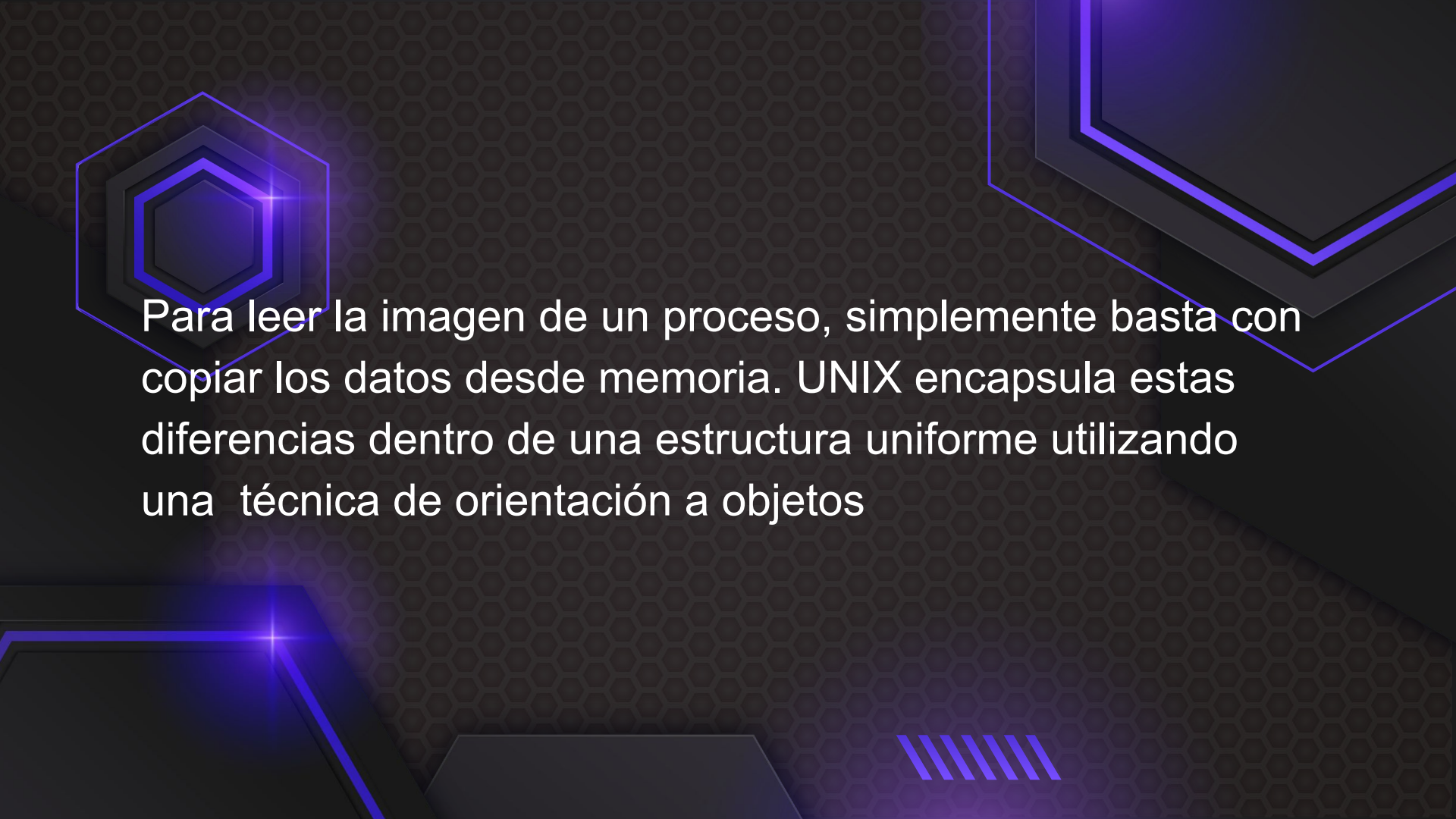
El *kernel* necesita mantener información de estado acerca del uso de los componentes de E/S. El *kernel* hace esto mediante diversas estructuras de datos internas al *kernel*, como por ejemplo la tabla de archivos abiertos. El *kernel* utiliza muchas estructuras similares para controlar las conexiones de red, las comunicaciones con los dispositivos de tipo carácter y otras actividades de E/S





UNIX proporciona acceso de sistema de archivos a diversas entidades, como los archivos de usuario, los dispositivos sin formato y los espacios de direcciones de los procesos. Aunque cada una de estas actividades soporta una operación read (), la semantica difiere.

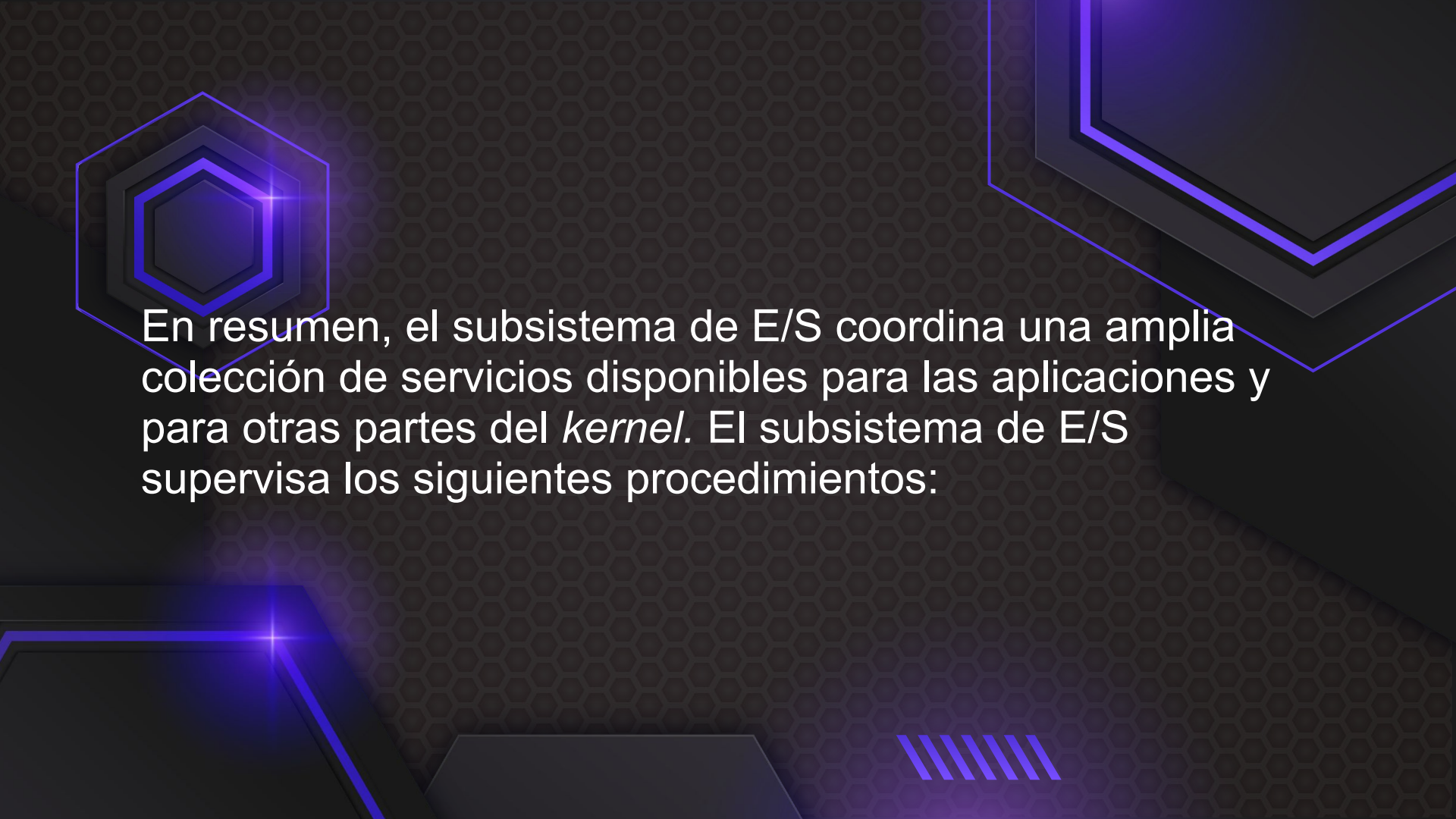




Para leer la imagen de un proceso, simplemente basta con copiar los datos desde memoria. UNIX encapsula estas diferencias dentro de una estructura uniforme utilizando una técnica de orientación a objetos


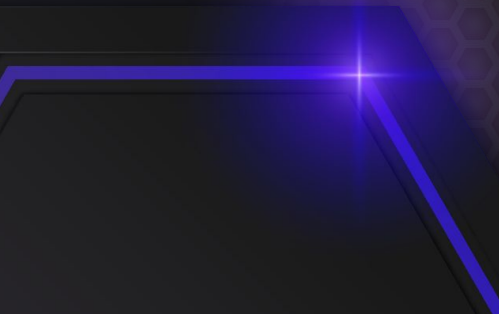


13.4.8 Resumen del subsistema de E/S del kernel





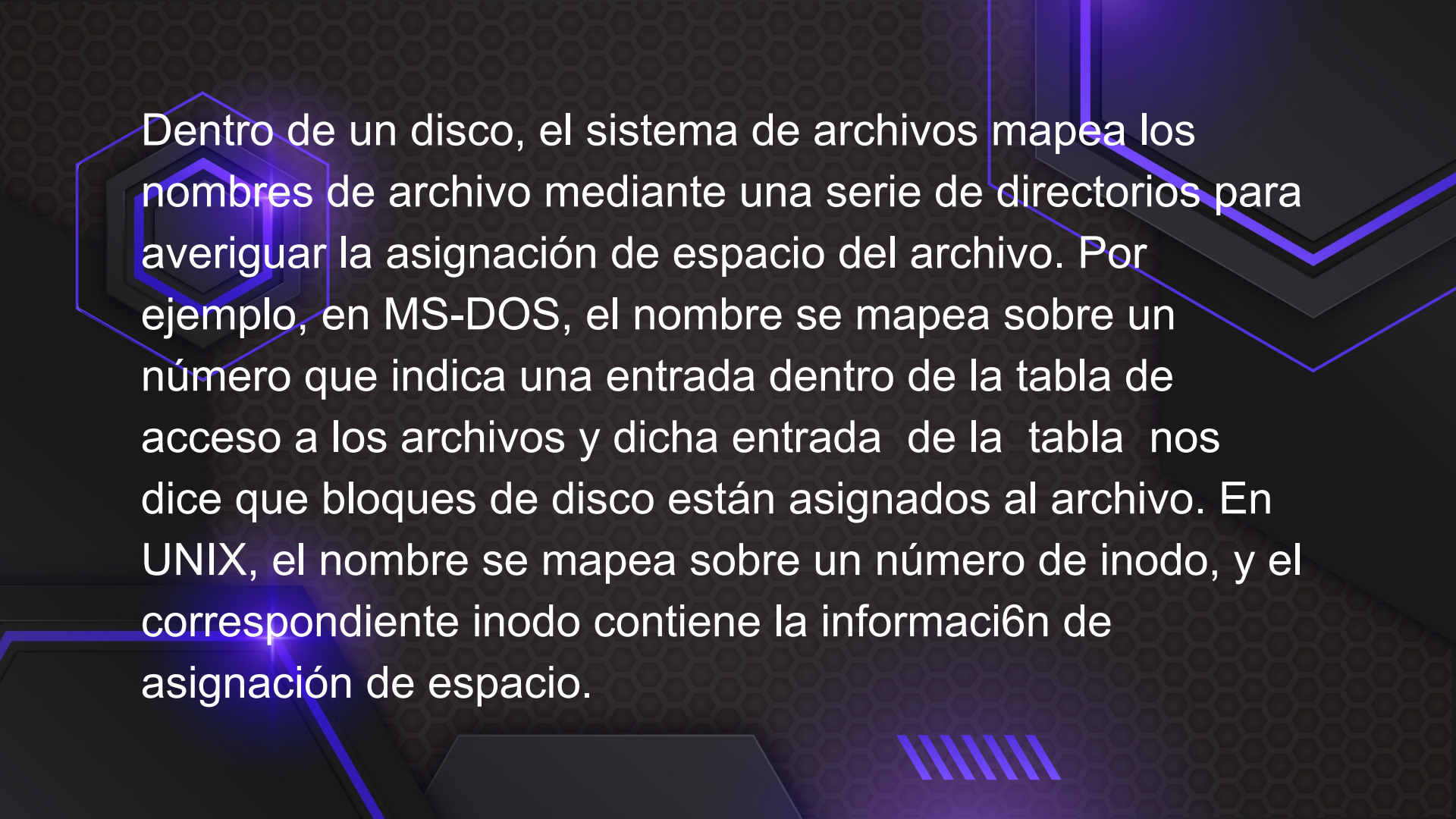
En resumen, el subsistema de E/S coordina una amplia colección de servicios disponibles para las aplicaciones y para otras partes del *kernel*. El subsistema de E/S supervisa los siguientes procedimientos:

- Gestión del espacio de nombres para archivos y dispositivos.
- Control de acceso a los archivos y dispositivos.
- Control de operaciones (por ejemplo, un módem no puede ejecutar una operación seek ()).
- Asignación de espacio en el sistema de archivos.
- Asignación de dispositivos.
- Almacenamiento en buffer.
- Almacenamiento en caché y gestión de colas de impresión.

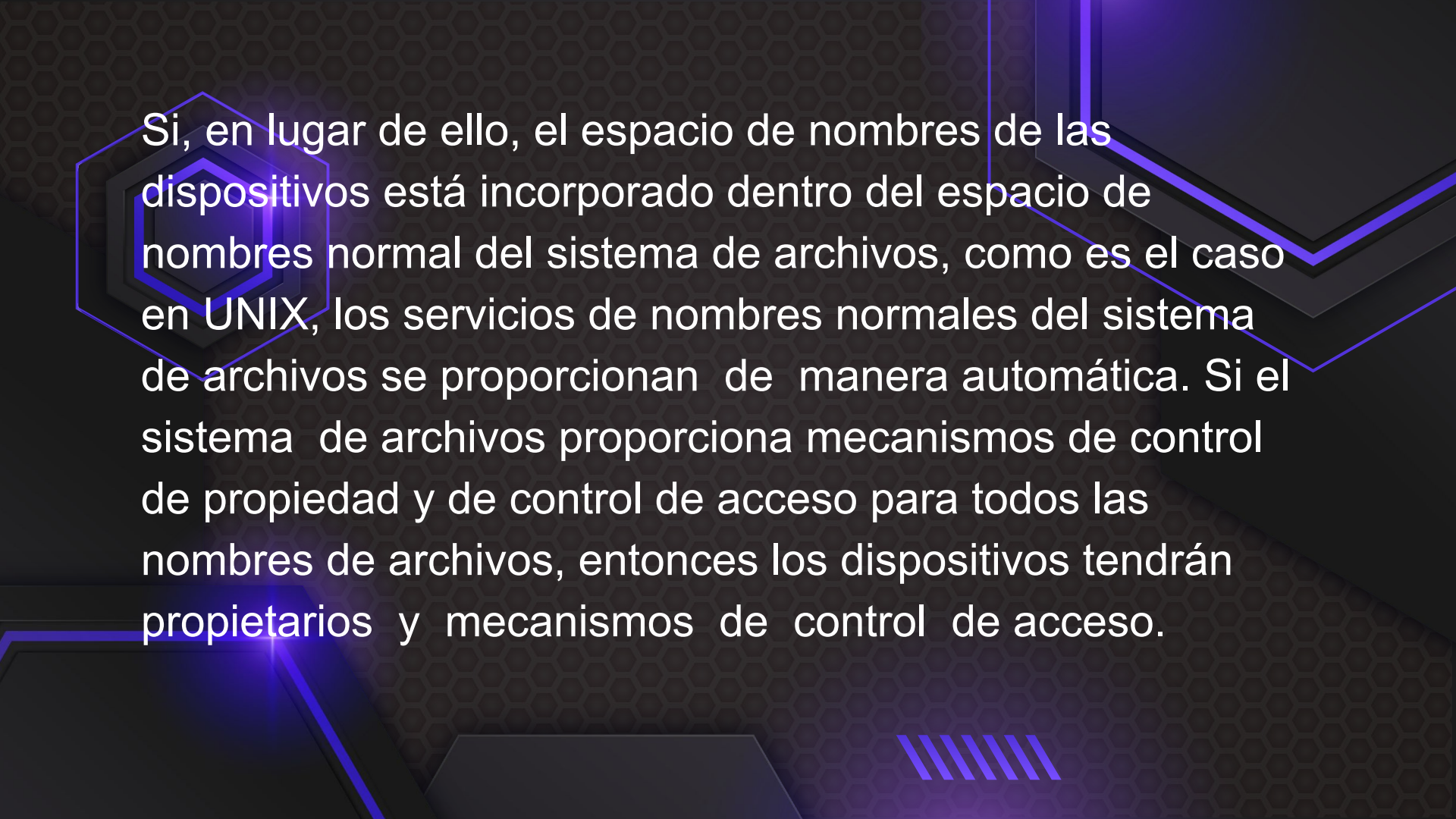
- 
- Planificación de E/5.
 - Monitorización del estado de los dispositivos, tratamiento de errores y recuperación de fallos.
 - Configuración e inicialización de controladores de dispositivos.
- 
- 
- 

13.5 Transformación de las solicitudes de E/S en operaciones hardware



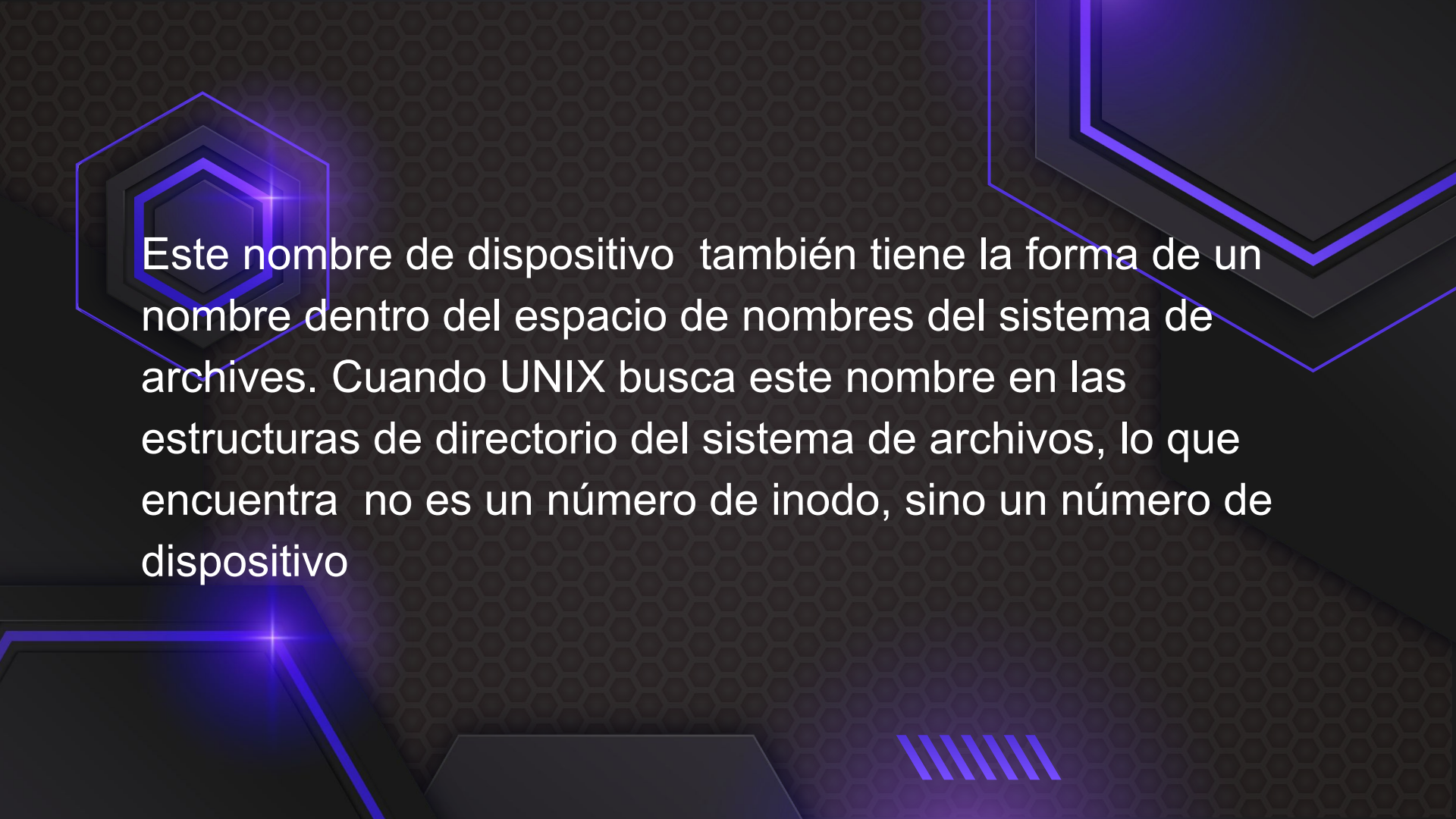


Dentro de un disco, el sistema de archivos mapea los nombres de archivo mediante una serie de directorios para averiguar la asignación de espacio del archivo. Por ejemplo, en MS-DOS, el nombre se mapea sobre un número que indica una entrada dentro de la tabla de acceso a los archivos y dicha entrada de la tabla nos dice que bloques de disco están asignados al archivo. En UNIX, el nombre se mapea sobre un número de inodo, y el correspondiente inodo contiene la información de asignación de espacio.

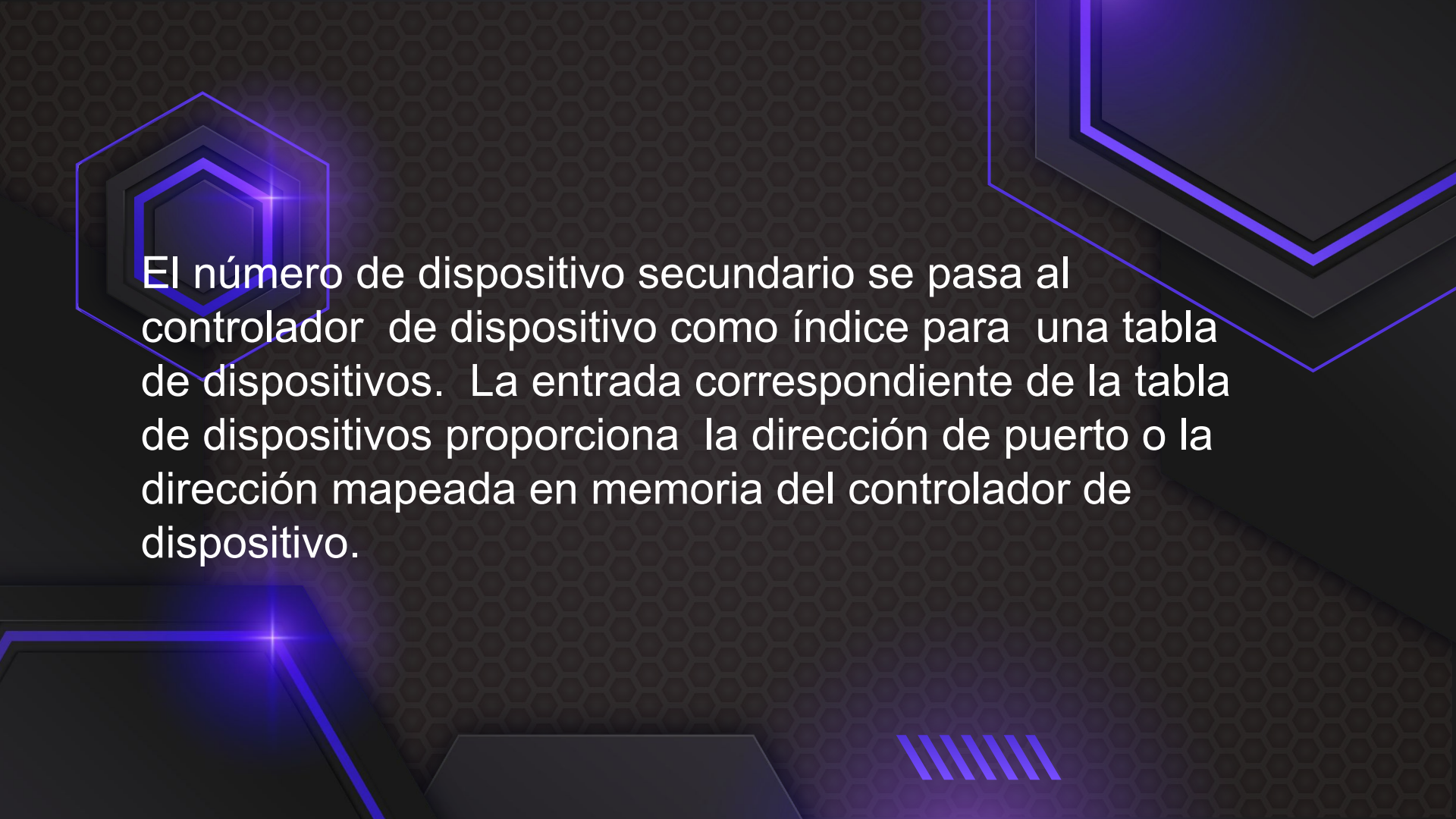


Si, en lugar de ello, el espacio de nombres de los dispositivos está incorporado dentro del espacio de nombres normal del sistema de archivos, como es el caso en UNIX, los servicios de nombres normales del sistema de archivos se proporcionan de manera automática. Si el sistema de archivos proporciona mecanismos de control de propiedad y de control de acceso para todos los nombres de archivos, entonces los dispositivos tendrán propietarios y mecanismos de control de acceso.


UNIX representa los nombres de los dispositivos dentro del espacio de nombres normal del sistema de archivos. A diferencia de un nombre de archivo MS-DOS, que tiene un separador de dos puntos, un nombre de ruta UNIX no tiene una separación clara de la parte correspondiente al dispositivo. De hecho, ninguna parte concreta del nombre de ruta es el nombre de un dispositivo. UNIX tiene una **tabla de montaje** que asocia prefijos de las nombres de ruta con nombres de dispositivo específicos.



Este nombre de dispositivo también tiene la forma de un nombre dentro del espacio de nombres del sistema de archivos. Cuando UNIX busca este nombre en las estructuras de directorio del sistema de archivos, lo que encuentra no es un número de inodo, sino un número de dispositivo



El número de dispositivo secundario se pasa al controlador de dispositivo como índice para una tabla de dispositivos. La entrada correspondiente de la tabla de dispositivos proporciona la dirección de puerto o la dirección mapeada en memoria del controlador de dispositivo.



El ciclo típico de vida de una solicitud de lectura bloqueante

1. Un proceso ejecuta una llamada al sistema bloqueante `read ()` dirigida a un descriptor de archivo o a un archivo que se haya abierto anteriormente.
2. El código de la llamada al sistema en el *kernel* comprueba la corrección de los parámetros. En el caso de una operación de entrada, si los datos ya están disponibles en la cache de buffer, los datos se devuelven en el proceso y se completa la solicitud de E/S.

3. En caso contrario, es necesario realizar una E/S física.
4. El controlador de dispositivo asigna espacio de buffer del *kernel* para recibir los datos y planifica la E/S.
5. La controladora del dispositivo opera el hardware del dispositivo para realizar la transferencia de datos.
6. El controlador puede realizar un sondeo para ver la información de estado y recolectar los datos, o puede haber configurado una transferencia de DMA hacia la memoria del *kernel*.

7. La rutina correcta de tratamiento de interrupciones recibirá la interrupción a través de la tabla de vectores de interrupción, almacenará los datos necesarios, efectuará una señalización dirigida al controlador de dispositivo y volverá de la interrupción.
8. El controlador de dispositivo recibe la señal, determina que solicitud de E/S se ha completado, determina el estado de la solicitud y señala al subsistema de E/S de! *kernel* que la solicitud se ha completado.

9. El *kernel* transfiere los datos a las códigos de retomo al espacio de direcciones del proceso solicitante y mueve el proceso de la cola de espera a la cola de procesos preparados.
10. Al mover el proceso a la cola de procesos preparados se desbloquea el proceso. Cuando el planificador asigna la CPU al proceso, este reanudará su ejecución en el punto correspondiente a la terminación de la llamada al sistema.