Sistemas Operativos 2

Unidad 2: Administración de dispositivos de E/S

Dispositivos especiales

René Ornelis
Primer semestre de 2025

Contenido

Índice de figuras

Dispositivos especiales

""

1 Introducción

Aunque existe una clasificación de dispositivos y una abstracción de los componentes generales de un dispositivo, existen dispositivos que no se les podría clasificar fácilmente dentro de estas categorías por sus características especiales. En este capítulo veremos los dispositivos que, por sus funcionalidades únicas, tienen un tratamiento especial y no pueden utilizarse como el resto de los dispositivos. La existencia de estos dispositivos hace difícil aplicar el principio de independencia del dispositivo, por lo que la mayoría de los fabricantes de sistemas operativos optan por aplicar un API especial a cada uno de estos.

Los dispositivos que estudiaremos son:

1. El reloj
2. La terminal
3. La red

2 El reloj

El concepto de reloj tiene varias acepciones en el ámbito de una computadora:

- Señal de reloj del procesador
- Temporizadores
- Reloj del sistema

2.1 Señal de reloj del procesador

Los procesadores cuentan con un circuito temporizador que genera señal periódica llamada el reloj o reloj del sistema, el cual es la base de todas la microoperaciones del procesador y determina la velocidad de este. Este circuito normalmente está fuera del ámbito del sistema operativo y lo único que se puede hacer es **acelerar al procesador** (*overclocking*), pero esto se realiza con herramientas específicas para cada procesador.

2.2 Reloj del sistema

También denominado reloj BIOS, es el reloj digital que mantiene la fecha y hora en el sistema. Este reloj es alimentado por batería, la cual se recarga cuando la computadora está encendida, por lo que en caso de que una computadora pasa mucho tiempo sin encenderse, al inicio puede solicitar establecer la fecha y hora.

Al encender la computadora, una de las primeras tareas del sistema operativo es leer la fecha y hora del BIOS y guardarla para posteriormente mantener el control (ver siguiente sección).

2.3 Temporizadores

Todo procesador tiene un conjunto de registros temporizadores, que tienen la capacidad de generar interrupciones basado en el valor de tiempo que se les asigne. Uno de estos temporizadores es la **interrupción de tiempo** o **interrupción del sistema**, el cual está conectado a una línea de interrupción de hardware de alta prioridad, y se puede programar en determinada frecuencia para que el sistema operativo pueda realizar determinadas tareas.

La frecuencia con que se programe la interrupción de tiempo depende del sistema operativo y del hardware. Por ejemplo: Linux programa la frecuencia en 100 y 1000 hz. El sistema operativo, basado en el conocimiento del procesador debe elegir esta frecuencia cuidando tener un balance adecuado entre la exactitud de sus cálculos y el rendimiento del sistema. Así, si se tiene una frecuencia alta de interrupción, como 1000 veces por segundo, los controles que haga el sistema en cada interrupción serán más precisos, pero tendrán un impacto directo en el rendimiento del sistema.

Las tareas usuales que realiza el sistema operativo en la interrupción del sistema son:

- Mantenimiento de la hora del sistema
- Muestreo del uso de procesador para estadísticas

2.3.1 Mantenimiento de fecha y hora

El sistema operativo no depende del reloj del BIOS para controlar la fecha y del sistema. Solamente en el arranque del sistema, se lee el reloj del BIOS, lo almacena en memoria y en la interrupción del sistema lo incrementa según el valor de la frecuencia de interrupción. De esta forma el sistema operativo mantiene el control de la fecha y hora. Al apagar el sistema, el valor de la fecha y hora se actualiza de vuelta al reloj del BIOS.

La mayoría de sistemas operativos mantienen el reloj del BIOS en el horario universal coordinado (UTC), por lo que, para mantener la hora del sistema, se debe tomar en cuenta la configuración de zona horaria y políticas de cada país respecto al horario de verano o invierno.

La forma de calcular el tiempo es normalmente un número entero con la cantidad de segundos, milisegundo u otra unidad, a partir de una fecha determinada. Por ejemplo: UNIX maneja la cantidad de segundos desde el 01-01-1970 00:00. Otras variantes de UNIX usan la cantidad de milisegundos desde la misma fecha. En el caso almacenar segundos, una variable de 32 bits alcanza para 136 años. Si utilizamos una variable de 64 bits, se pueden almacenar más de 586 mil millones de años. Por otro lado, Windows almacena la fecha en centenas de nanosegundos (1/10,000,000), lo que permite más 58,000 años en una variable de 64 bits. El valor de esta variable es la que sirve de base para calcular la fecha y hora del sistema en el calendario gregoriano o cualquier otro calendario que se tenga configurado, lo cual es una operación relativamente complicada.

Adicional al mantenimiento automático, el sistema operativo también debe permitir al administrador cambiar la hora del sistema, lo cual se realizar de dos formas:

- Interfaz para cambiar la fecha y hora, lo cual usualmente solo se da acceso al superusuario.
- Actualización a través de servicios de red (protocolo ntp)

En ambos casos, el administrador debe tener cuidado, ya que el cambio de fechas muy adelante o muy hacia atrás, puede provocar efectos indeseados debido principalmente a tareas calendarizadas, que se dispararían muchas al mismo tiempo si se adelante la hora, o que ya no se realizarán si se atrasa. También puede haber implicaciones en vencimientos de licencias o de programas que están programados con codificación fija para funcionar en determinado período de tiempo.

Es por esto por lo que **ntp** no permite la sincronización si la diferencia entre la hora del servidor y la hora de la computadora es más de 1000 segundos (configurable). En todo caso, si una computadora se desvía mucho de la hora (por ejemplo, si pasó mucho tiempo apagada), lo mejor es hacer cambios incrementales hasta tener la hora exacta.

2.3.2 Estadísticas.

Otras de las funciones en que la interrupción del sistema ayuda es en la obtención de estadísticas de uso del sistema. Por ejemplo: el porcentaje de uso del procesador, de forma simplificada, se obtiene así:

1. La rutina de interrupción usa dos variables: número de muestras y número de esas muestras en la que el procesador estaba en uso. En cada interrupción, el número de muestra se incrementa incondicionalmente y si hay un proceso que esté en uso del procesador (durante la interrupción) entonces se incrementa la segunda variable.
2. Otros procesos se despiertan periódicamente para revisar la información y realizar los cálculos correspondientes. En este caso, el porcentaje de uso del procesador se calcula así:

$$Porcentaje\ de\ uso\ del\ procesador = \frac{Muestras\ con\ uso\ de\ procesador}{Número\ de\ muestras\ tomadas}$$

Claro que la implementación es un poco más complicada, ya que este porcentaje se determina por proceso y, además del uso del procesador se controlan otros indicadores como el uso del disco duro, uso de la red, etc. Sin embargo, el punto es que la rutina de interrupción no realiza los cálculos de las estadísticas, sino que solo cuenta los eventos y son otros procesos de fondo del kernel los que se encargan de hacer los cálculos correspondientes, ya que la rutina de interrupción debe ser los más breve posible.

2.3.3 Gestión de temporizadores

Además de la interrupción del sistema por medio del registro temporizador de máxima prioridad, los procesadores suelen tener más registros temporizadores los cuales se utilizan con menor prioridad, pero nos sirven para las necesidades de temporización de los procesos como las agendas con avisos, los tiempos de espera de la red, cronómetros, alarmas, etc.

El sistema operativo logra atender las necesidades de temporización de todos los procesos a través del manejo de una cola de requerimientos (ver Figura 1), la cual está ordenada por la hora en la que el proceso necesita ser notificado. Los requerimientos de los procesos se muestran en hora, minutos, segundos y milisegundos, solo con fines didácticos, ya que usualmente el valor requerido se suele expresar en milisegundos.

Suponiendo que solo existe un registro temporizador, en el ejemplo se muestra que el requerimiento actual la hora el sistema es la 09:32:05.1527 y se atenderá una temporización para las 09:35:25, lo cual significa 248,473 milisegundos.

Figura 1: Cola del temporizador

Cuando el temporizador llegue a cero, el sistema notificará al proceso que hizo el requerimiento y tomará el próximo requerimiento de la cola, que corresponde a las 09:40:34, lo que en su momento corresponderá a 309,000 milisegundo (el lector verifique el cálculo), y se repetirá el proceso.

Si el procesador cuenta con más de un registro temporizador, este mismo concepto se puede expandir a manejar una cola para varios temporizadores, tal como se muestra en la Figura 2, donde el primer temporizador que finalice su requerimiento tomará el siguiente elemento de la cola y lo atenderá.

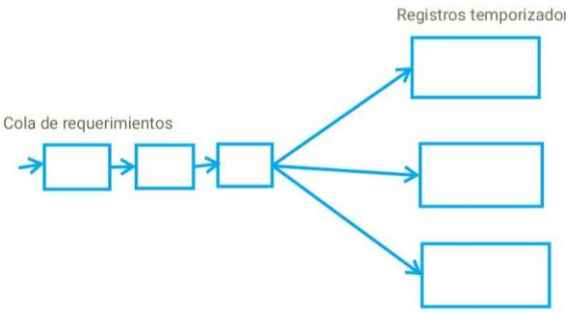

Figura 2: Cola para múltiples temporizadores

Esto tiene la ventaja que se aprovecha al máximo todos los temporizadores existentes ya que mientras haya requerimientos en cola, los registros temporizadores estarán trabajando.

Por otro lado, puede ser que el sistema operativo implemente políticas orientadas a especializar los registros temporizadores en ciertas tareas. Por ejemplo: un registro temporizador para las tareas del kernel, otro para las necesidades de hardware (como espera de red, discos, etc.) y otro para atender los requerimientos de los procesos de usuario. En este caso, se deberá manejar una cola por temporizador según se muestra en la Figura 3, de tal forma que cada temporizador tiene su propia cola de requerimientos.

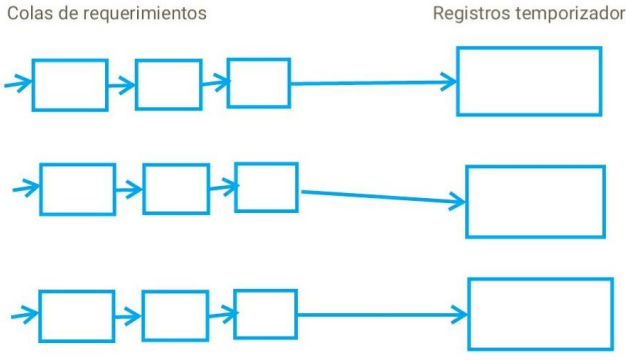

Figura 3: Múltiples colas de temporización

Para que los procesos puedan realizar las operaciones con temporizadores, el sistema operativo debe proporcionar una API especial el cual contiene las operaciones necesarias. En POSIX (la especificación de los UNIX y LINUX), la gestión de temporizadores está definida con las siguientes cinco primitivas:

Nombre	Descripción
Timer_create	Crea un temporizador
Timer_delete	Elimina un temporizador
Timer_getoverrun	Cuenta de desbordamiento
Timer_gettime	Tiempo restante
Timer_settime	Programa y arranca el temporizador

2.3.4 Soporte para la planificación de procesos

El planificador utiliza el reloj del sistema para determinar el tiempo adecuado para otorgar y quitar el procesador a cada proceso. Una parte del trabajo se realiza en la rutina de interrupción donde solo se cuentan los "*ticks*" que lleva activo el proceso que está en uso del procesador. Por otra parte, los otros procesos de planificación se encargarán de definir si al proceso activo le corresponde quitarle el procesador y dárselo a otro proceso, luego de tomar en cuenta el tiempo que lleva activo, la prioridad y otras consideraciones de la política de planificación.

Tradicionalmente la planificación de procesos y estadísticas se ha realizado a través de la interrupción periódica del reloj, o resolución del reloj (100hz-1000hz), pero en el kernel 2.6.21 de linux ya se implementa a través de temporizadores (***dynticks= dynamic ticks***), lo que supone mejoras en uso del CPU para ahorro de energía y optimización dentro de un hipervisor.

3 La terminal

Históricamente se le denomina terminal a la combinación de teclado y pantalla, pues las primeras terminales eran un solo dispositivo serial (terminal tonta) que recibía entradas y salida y se proyectaba en pantalla. Recordemos que estos solían ser pantallas de modo texto donde no existía el posicionamiento del cursor sino solamente el despliegue secuencial de información. Actualmente los sistemas UNIX conservan el dispositivo */dev/ttyX* los cuales se tratan como terminales que se pueden utilizar para la pantalla + teclado o para sesiones remotas.

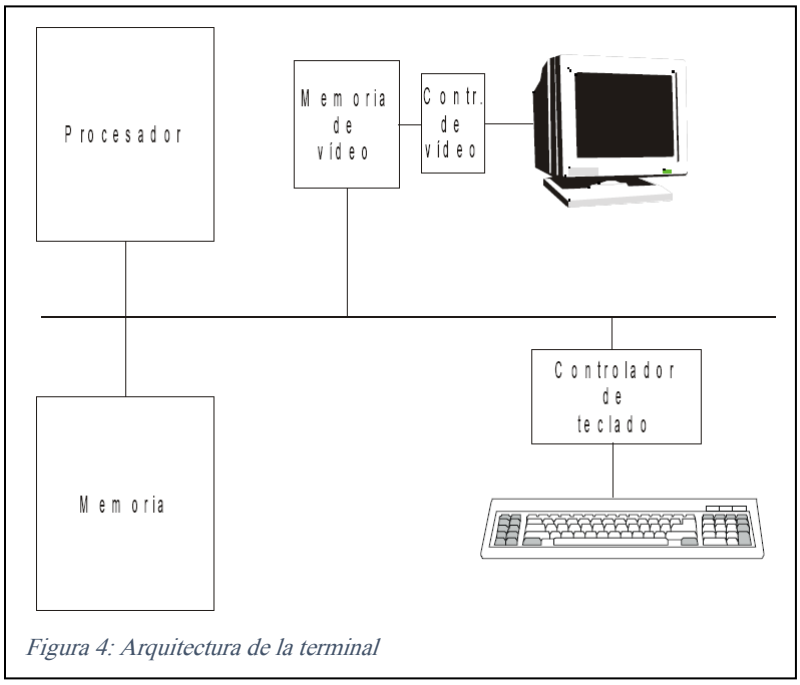

Figura 4: Arquitectura de la terminal

Tal como se puede apreciar en la Figura 4, actualmente tanto el teclado como la pantalla se tratan separadamente. El teclado se ve como un dispositivo de interfaz, de carácter, proyectado en

puertos, mientras que la pantalla es un dispositivo de interfaz, de bloques, proyectado en memoria.

3.1 El teclado

- Teclado genera interrupción al pulsar tecla
  - S.O. lee código de tecla de registro de controlador de teclado
  - Conversión a ASCII y manejo de teclas modif. por SW
  - Manejador proporciona "tecleado anticipado" (type ahead)
  - Usuario teclea info. antes de que programa la solicite
  - Manejador debe usar zona de almacenamiento intermedio

El teclado funciona principalmente como un dispositivo de entrada, pero puede ser considerado de salida en algunos contextos más específicos. Tal como se explica a continuación:

3.1.1 Teclado como dispositivo de entrada

Cuando se usa un teclado, se está introduciendo información al sistema operativo. El proceso general es el siguiente:

1. **Presión de teclas:** Cuando presionas, se suelta o se mantiene presionada una tecla, se activa un interruptor (en teclados mecánicos) o se envía una señal (en teclados de membrana o de láminas).
2. **Generación de señales:** Cada tecla está asociada a un código único conocido como código de escaneo o código de clave (scan code). Este código es enviado al controlador del teclado. Estas señales incluyen el estado de las teclas modificadoras con alt, control y shift.
3. **Transmisión al Sistema Operativo**: El controlador del teclado envía el código de la tecla presionada al sistema operativo a través del bus del sistema. En muchos casos, esto sucede a través de un puerto USB o un conector PS/2.
4. **Recepción por el Sistema Operativo:** El sistema operativo recibe el código de la tecla, a través de la rutina de servicio de interrupción (que es parte del driver), y lo ingresa a una cola o buffer del teclado.
5. **Acción del Sistema Operativo**: Finalmente, el sistema operativo toma los códigos en la cola de ingreso, lo interpreta como un carácter o comando y pasa esta información a la aplicación activa o al programa que está ejecutando, que realiza la acción correspondiente (como mostrar el carácter en una pantalla o ejecutar un comando). La interpretación del *scan code* depende de varias configuraciones como la disposición y lenguaje del teclado.

3.1.2 Teclado como dispositivo de salida

Aunque el teclado es principalmente un dispositivo de entrada, hay casos donde también puede considerarse como un dispositivo de salida:

1. **Indicadores y retroalimentación**: Algunos teclados tienen luces indicadoras (como las luces de Bloq Mayús, Bloq Num y Bloq Despl) que informan al usuario sobre el estado

de ciertas funciones. Estas luces son controladas por el sistema operativo y el firmware del teclado.

2. **Teclados especializados**: Algunos teclados avanzados o programables pueden tener pantallas pequeñas, retroiluminación personalizada, o teclas que muestran información dinámica. En estos casos, el teclado puede enviar información de salida visual al usuario en función de la interacción con el sistema operativo o las aplicaciones.

3.2 El monitor

La pantalla (o monitor) funciona como un dispositivo salida, de bloque y proyectado en memoria en un sistema operativo. La pantalla puede operar en modo texto o en modo gráfico.

En ambos modos, el monitor actúa como un dispositivo de salida, pero en el modo gráfico también puede integrar funcionalidades de entrada, especialmente en dispositivos táctiles.

La pantalla es un dispositivo proyectado en memoria (la memoria de video), lo cual significa que el sistema operativo, a través del *driver*, escribe en la memoria de video y la controladora transforma la información de la memoria de video en las señales electrónicas para que la pantalla despliegue la información correspondiente. Tal como se muestra en la Figura 5, el procesamiento es así:

1. Un proceso especifica al sistema operativo, a través del API, que desea escribir en pantalla (en este ejemplo la letra "A")
2. Basado en las configuraciones de fuente (*font*), tamaño de letra y resolución, el sistema operativo convierte el requerimiento en un mapa de pixeles que represente la letra solicitada y lo escribe en la memoria de video. En este ejemplo en pixeles de color blanco (**b**) y negro (**N**).
3. La tarjeta controladora lee periódicamente la información de la memoria de video y envía las señales electrónicas al monitor para que se despliegue la información correspondiente.

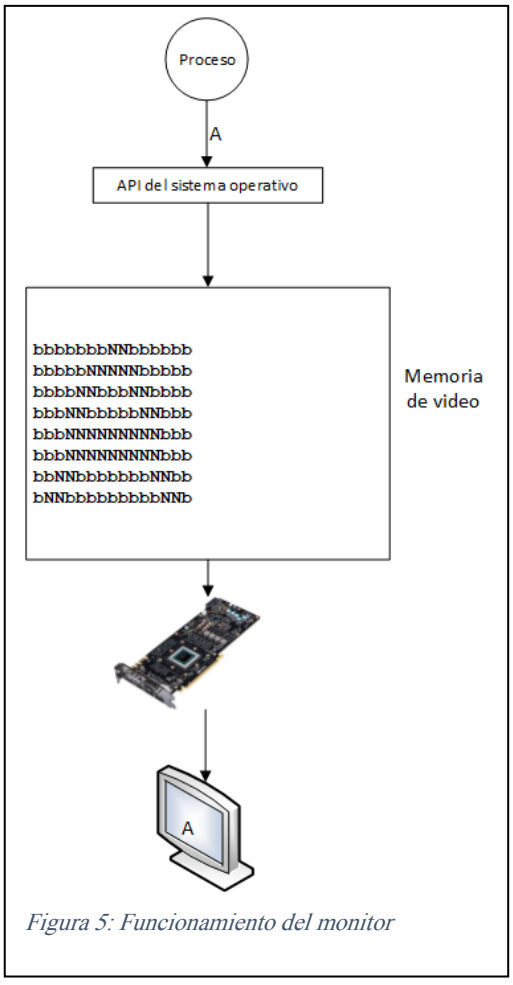

Figura 5: Funcionamiento del monitor

Usualmente, la tarjeta controladora tiene una **tasa de refrescamiento** (*refresh rate*) entre 60 y 80 hz, es decir, se lee la memoria de video entre 60 y 80 veces por segundo. Actualmente, las controladoras de alta resolución pueden tener hasta 250 hz.

3.2.1 Modo texto

En el modo texto, el monitor muestra información en una forma de texto estructurada en líneas y columnas. No se usan gráficos complejos ni imágenes, solo caracteres alfanuméricos y símbolos. Cada carácter se representa en una celda de una cuadrícula (matriz de caracteres) la cual depende de la resolución del modo texto en términos del número de filas y columnas de caracteres (por ejemplo, 80x25). Las interfaces en modo texto son más simples y menos exigentes en términos de recursos gráficos.

En modo texto, la memoria de video del sistema se organiza en una tabla que contiene los códigos de los caracteres a mostrar. Cada entrada en esta memoria de video representa un carácter que debe ser mostrado en una posición específica en la pantalla. El código de cada posición, además del código del carácter a representar, también incluye el código de color de fondo y frente del carácter.

Cuando una aplicación solicita al sistema operativo que se muestre texto en la pantalla, este escribe los códigos de los caracteres en la memoria de video, la cual es leída por el controlador y convierte los códigos de los caracteres en representaciones gráficas (patrones de pixeles) que se muestran en la pantalla. Cada carácter se muestra en una celda de la cuadrícula de la pantalla.

Muchos sistemas operativos usan el modo texto en sus terminales y consolas para mostrar información y permitir al usuario interactuar mediante comandos de texto.

3.2.2 Modo gráfico

En el modo gráfico, el monitor muestra una matriz de pixeles, donde cada posición contiene el código de color del pixel a mostrar.

Cuando una aplicación solicita al sistema operativo que se muestre texto en la pantalla, este debe transformar cada carácter en un mapa de bits que represente dicho carácter y lo escribe en la memoria de video. Esta trasformación es determinada por configuraciones como el font, el tamaño de cada carácter y la resolución de la pantalla.

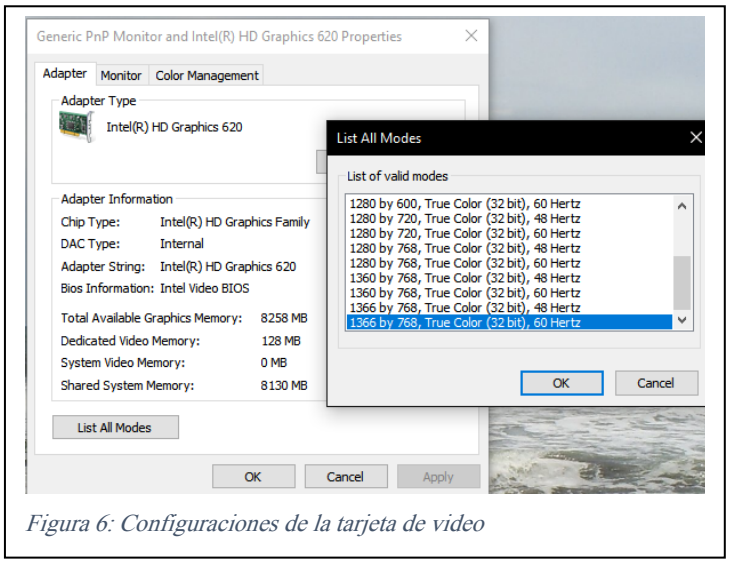

Figura 6: Configuraciones de la tarjeta de video

La resolución de una pantalla (ver Figura 6) depende de dos factores:

1. Matriz de pixeles: cuántos pixeles se manejan de ancho y alto en el monitor.

2. Bits de color: cantidad de bits que se utilizan para representar un color.

Entre más altos sean estos valores, mayor será la nitidez de la pantalla, pero también implica que se utilizará más memoria. Por ejemplo: en la Figura 6 se tiene configurado una matriz de 1366 x 768 pixeles y cada pixel ocupa 32 bits, lo cual implica que se necesitan 4Mb de memoria de video.

3.2.3 Dispositivo de entrada

Aunque la pantalla se usa principalmente como salida, algunas pantallas modernas también actúan como dispositivos de entrada, especialmente las pantallas táctiles. En estos casos las pantallas táctiles pueden detectar la ubicación y el tipo de interacción del usuario (toque, deslizamiento, pellizco, etc.) a través de diversas tecnologías, como capacitiva o resistiva.

Cuando el usuario toca la pantalla, los datos de la posición del toque se envían al sistema operativo. El sistema operativo interpreta estas señales y las traduce en comandos para la aplicación en uso.

4 La red

Dispositivos de comunicación / de caracter

- Dada su creciente importancia, soporte de S.O. cada vez mejor
- API especial para la red (SOCKETS), debido a varios factores:
    - Direcciones de otras máquinas
    - Puertos de red
    - Protocolo de comunicación
    - Timeout
    - Retry

    El software de red está organizado en tres niveles:

    - Nivel de interfaz a las aplicaciones
        - Típicamente, sockets (Winsock en Windows)
        - Puede considerarse como nivel de sesión OSI
    - Nivel de protocolos
        - Capa(s) que implementa(n) transporte y red OSI (o TCP/IP)
        - Incluye funciones de encaminamiento
    - Nivel de dispositivo de red
        - Manejadores de dispositivos de red (nivel de enlace OSI)

4.1 Sockets

Sockets: se basa en las funciones que se muestran en la Figura 7.

Los estados por los que puede pasar un socket se ilustran en la Figura 8.

Primitives	Meaning
SOCKET	Create a New Communication Endpoint.
BIND	Attach a Local Address to a SOCKET.
LISTEN	Shows the Willingness to Accept Connections.
ACCEPT	Block the Caller until a Connection Attempts Arrives.
CONNECT	Actively Attempt to Establish a Connection.
SEND	Send Some Data over Connection.
RECEIVE	Receive Some Data from the Connection.
CLOSE	Release the Connection.

Figura 7: API para sockets

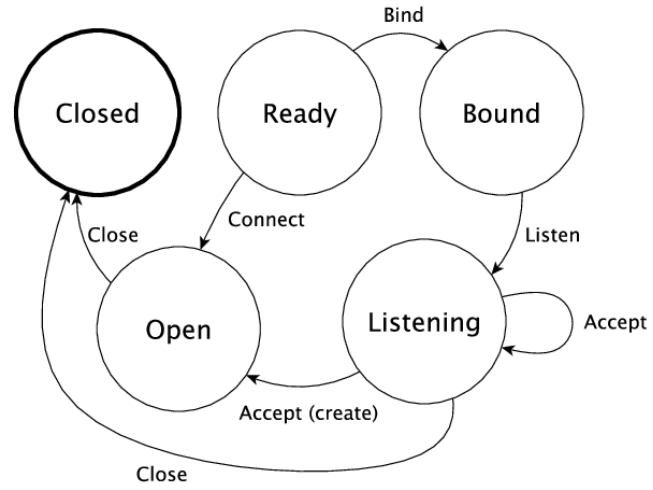

Figura 8: Estados de un socket