Administración de E/S

El sistema operativo debe:

- Emitir comandos para los dispositivos
- Captar interrupciones
- Manejar errores
- Proporcionar una interfaz "simple y fácil de usar" entre los dispositivos y el resto del sistema. Si es posible, que sea igual para todos los dispositivos. Independencia de dispositivos

Los dispositivos externos dedicados a la E/S en un computador se pueden agrupar en **tres categorías**:

- **Legibles para el usuario**: Adecuados para la comunicación con el usuario del computador. (Ej: Impresora y terminales gráficas)
- **Legibles para la máquin**a: Adecuados para la comunicación con equipamiento electrónico. (Ej: Unidades de disco y sensores)
- **Comunicación**: Adecuados para la comunicación con dispositivos remotos. (Ej: Controladores y módems)

Hay grandes **diferencias** entre las distintas categorías e incluso dentro de cada una:

- Velocidad de transferencia de datos.
- Aplicación
- Complejidad de control.
- Unidad de transferencia.
- Representación de datos
- Condiciones de error

Los **dispositivos de E/S** se pueden dividir básicamente en **dos categorías**:

- **Dispositivos de bloque**:
  - Almacena información en bloques de tamaño fijo, cada uno con su propia dirección.
  - Todas las **transferencias se realizan** en unidades de uno o más **bloques completos** (consecutivos).
  - La propiedad esencial de un dispositivo de bloque es que **es posible leer o escribir cada bloque de manera independiente** de los demás.
  - **Son direccionables**.
  - **Permiten búsquedas**.
- **Dispositivos de carácter**:
  - Envía o acepta un flujo de caracteres, sin importar la estructura del bloque.
  - **No es direccionable**
  - **No tiene ninguna operación de búsqueda**.

Controladores de dispositivos

Las unidades de E/S consisten en un **componente mecánico**, es decir, **el dispositivo en sí**. Un **componente electrónico**, llamado **controlador del dispositivo** o adaptador, que tiene forma de chip o tarjeta de circuito integrado. Y una **interfaz estandarizada** entre el componente mecánico y electrónico, la cual normalmente es de muy bajo nivel y permite a las empresas fabricar controladores que se adapten a esa interfaz.

FUNDAMENTOS DEL SOFTWARE DE E/S

Ahora vamos a alejarnos del hardware de E/S para analizar el software de E/S.

Objetivos del software de E/S

Dentro de los objetivos del software de E/S existen distintos conceptos que debemos tener en cuenta:

- **Independencia de dispositivos**: Debe ser posible escribir programas que puedan acceder a cualquier dispositivo de E/S sin tener que especificar el dispositivo por adelantado.
  Por ejemplo, un programa que necesita leer un archivo debe tener la capacidad de hacerlo sin especificar que es un CD-ROM, DVD o USB. **Depende del sistema operativo encargarse de los problemas** producidos por el hecho **de que estos dispositivos sean diferentes**.
- **Denominación uniforme**: El nombre de un archivo o dispositivo simplemente debe ser una cadena o un entero sin depender del dispositivo de ninguna forma. En UNIX por ejemplo, **todos los archivos y dispositivos se direccionan mediante el nombre de una ruta**.
- **Manejo de errores**: Los errores se deben manejar **lo más cerca del hardware que sea posible**. Si el controlador descubre un error de lectura, debe tratar de corregir el error por sí mismo. Si no puede, entonces el software controlador del dispositivo debe manejarlo, tal vez con sólo tratar de leer el bloque de nuevo.
- **Transferencias síncronas (bloqueo) y asíncronas (interrupciones)**:
  - La mayoría de las operaciones de E/S son asíncronas, es decir, la **CPU inicia la transferencia** y **se va a hacer algo más** hasta que llega la interrupción.
  - Los programas de usuario son más fáciles de escribir si las operaciones de E/S son de bloqueo, es decir, d**espués de una llamada al sistema** de lectura, el **programa se suspende** de manera automática hasta que haya datos disponibles en el bufer.
- **Uso de búfer**: Normalmente, los datos que provienen de un dispositivo no se pueden almacenar directamente en su destino final, por lo que los datos se deben colocar en un búfer de salida por adelantado para desacoplar la velocidad a la que se llena el búfer, de la velocidad a la que se vacía, de manera que se eviten sub-desbordamientos de búfer.
- **Dispositivos compartidos y dedicados**:
  - Los dispositivos **compartidos** (como discos), pueden ser **utilizados por muchos usuarios a la vez sin producir problemas**.
  - Los dispositivos **dedicados** (como cintas), tienen que estar **dedicados a un solo usuario hasta que este termine**.

ORGANIZACIÓN DEL SISTEMA DE E/S

Existen 3 técnicas para llevar a cabo la E/S:

- **E/S programada**: El procesador envía un mandato de E/S, a petición de un proceso, a un módulo de E/S; a continuación, ese proceso realiza una espera activa hasta que se complete la operación antes de continuar.
  Ejemplo:

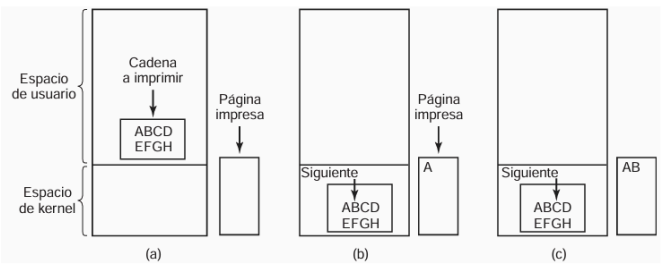


Figura 5-7. Pasos para imprimir una cadena.

Imagina un proceso de usuario que desea imprimir la cadena “ABCDEFGH” en una impresora. Primero, ensambla la cadena en un búfer en espacio de usuario. Luego, adquiere la impresora y hace una llamada al sistema para indicar que imprima la

cadena. El sistema operativo copia la cadena al espacio de kernel y verifica si la impresora está disponible. Si lo está, se copia el primer carácter al registro de datos de la impresora. A partir de ahí, el sistema operativo entra en un ciclo estrecho, imprimiendo los caracteres uno a la vez. El aspecto esencial de la E/S programada es que después de imprimir un carácter, la CPU sondea en forma continua el dispositivo para ver si está listo para aceptar otro. Este comportamiento se conoce comúnmente como sondeo u ocupado en espera. Sin embargo, este método es ineficiente en sistemas más complejos. Se necesita un mejor enfoque de E/S

- **E/S dirigida por interrupciones**: El procesador emite un mandato de E/S a petición de un proceso y continúa ejecutando las instrucciones siguientes (del mismo proceso u otro), siendo interrumpido por el módulo de E/S cuando éste ha completado su trabajo.

- **Acceso directo de memoria (DMA)**): Un módulo de DMA controla el intercambio de datos entre la memoria principal y un módulo de E/S.
  El procesador manda una petición de transferencia de un bloque de datos al módulo de DMA y resulta interrumpido sólo cuando se haya transferido el bloque completo.
  Ejemplo:
  Una desventaja de la E/S dirigida por interrupciones es que se necesita una por cada carácter y cada una de ellas requiere tiempo. La solución está en utilizar un DMA, el cual alimenta los caracteres a la impresora uno a la vez, sin que la CPU se moleste. En esencia, **el DMA es E/S programada, sólo que el controlador de DMA realiza todo el trabajo en vez de la CPU principal**.
  La gran ganancia con DMA es reducir el número de interrupciones, de una por cada carácter a una por cada búfer impreso.

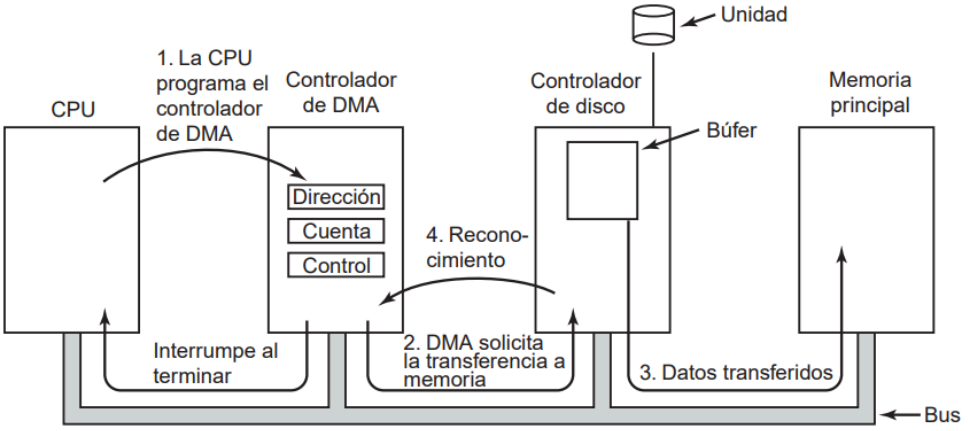


Figura 5-4. Operación de una transferencia de DMA.

Tabla 11.1. Técnicas de E/S.

	Sin interrupciones	Con interrupciones
Transferencia de E/S a memoria a través del procesador	E/S programada	E/S dirigida por interrupciones
Transferencia directa de E/S a memoria		Acceso directo a memoria (DMA)

LA EVOLUCIÓN DEL SISTEMA DE E/S

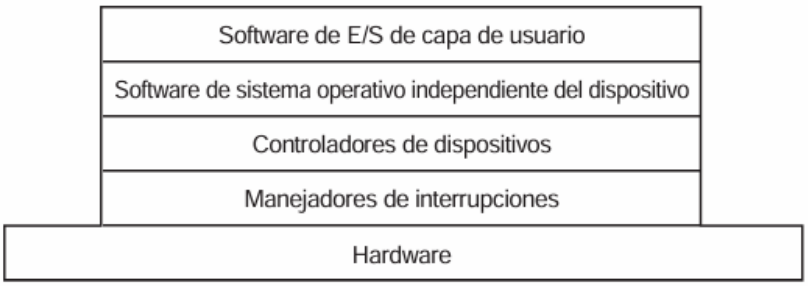
Según los computadores han ido evolucionando, ha habido una creciente complejidad y sofisticación de los componentes individuales, evidente también en las funciones de E/S. Las etapas de esta evolución se pueden resumir de la siguiente manera:

1. **El procesador controla directamente un dispositivo periférico**.
2. **Se añade un controlador o módulo de E/S**. El procesador usa E/S programada sin interrupciones. Con este paso, el procesador se independiza de los detalles específicos de las interfaces de los dispositivos externos.
3. **Se utiliza la misma configuración que en la etapa anterior, pero empleando interrupciones**. El procesador no necesita gastar tiempo esperando a que se realice una operación de E/S, incrementando de esta manera la eficiencia.
4. **Al módulo de E/S se le da control directo de la memoria mediante DMA**. Con ello, puede mover un bloque de datos a la memoria sin involucrar el procesador, excepto al principio y al final de la transferencia.
5. **Se mejora el módulo de E/S para convertirse en un procesador independiente, con un juego de instrucciones especializadas adaptadas a la E/S**. La unidad central de procesamiento (CPU) hace que el procesador ejecute un programa de E/S residente en la memoria principal, llamado **canal de E/S**. El procesador de E/S lee y ejecuta estas instrucciones sin la intervención del procesador. Esto permite que el procesador especifique una secuencia de actividades de E/S, siendo interrumpido sólo cuando se termine la secuencia completa.
6. **El módulo de E/S tiene su propia memoria local y es, de hecho, un computador por derecho propio**. Con esta arquitectura, se pueden controlar un gran conjunto de dispositivos de E/S, con una intervención mínima por parte del procesador. Un uso común de esta arquitectura ha sido controlar la comunicación con terminales interactivos. El procesador de E/S se encarga de la mayoría de las tareas involucradas en el control de los terminales.

En conclusión, cada vez hay una mayor parte de las tareas de E/S que se realizan sin la intervención del procesador, mejorando el rendimiento.

Capas del Software de E/S

El software de E/S se organiza en cuatro capas, cada una de ellas tiene una función bien definida que realizar, y una interfaz bien definida para los niveles adyacentes.



Manejadores de interrupciones

Las interrupciones son un hecho incómodo de la vida y no se pueden evitar. Estas deben ocultarse y la mejor manera para hacerlo es que el controlador que inicia una operación de E/S se bloquee hasta que se haya completado la E/S y ocurra la interrupción. Cuando ocurre la interrupción, el

procedimiento de interrupciones hace todo lo necesario para poder manejarlo. El efecto neto de la interrupción será que un controlador que estaba bloqueado podrá ejecutarse ahora. Este modelo funciona mejor si los controladores están estructurados como procesos del kernel, con sus propios estados, pilas y contadores del programa. El procesamiento de interrupciones no carece de importancia y ocupa un considerable número de instrucciones de la CPU.

Controladores de dispositivos

Vimos que cada controlador tiene ciertos registros de dispositivos que se utilizan para darle comandos o ciertos registros de dispositivos que se utilizan para leer su estado, o ambos. Estos varían de un dispositivo a otro.

Cada dispositivo de E/S conectado a una computadora necesita cierto código específico para controlarlo, llamado **driver**, es escrito por el fabricante del dispositivo y se incluye junto con el mismo. Cada uno de estos maneja un dispositivo o una clase de dispositivos (Normalmente solo 1).

Para poder utilizar el hardware del dispositivo, el driver por lo general tiene que **formar parte del kernel del S.O**, así funciona en arquitecturas actuales.
Aunque **es posible construir controladores que se ejecuten en el espacio de usuario**. Este diseño aísla al kernel de los controladores, y a un controlador de otro, eliminando una fuente importante de fallas en el sistema: controladores con errores que interfieren con el kernel de una manera u otra. En sistemas altamente confiables es la mejor forma de hacerlo.

La mayoría de los sistemas operativos definen una **interfaz estándar** que todos los controladores de bloque deben aceptar. Estas interfaces consisten en varios procedimientos que el resto del sistema operativo puede llamar para hacer que el controlador realice un trabajo para él. Los **procedimientos ordinarios** son los que se utilizan para **leer un bloque** (dispositivo de bloque) o **escribir una cadena de caracteres** (dispositivo de carácter)

Los sistemas operativos (empezando con MS-DOS) se inclinaron por un modelo en el que **el controlador se cargaba en forma dinámica en el sistema durante la ejecución.**
Un controlador de dispositivo tiene varias **funciones**:

- **Aceptar peticiones** abstractas **de lectura y escritura del software** independiente del dispositivo que está por encima de él, y ver que se lleven a cabo.
- Inicializar el dispositivo, si es necesario.
- Administrar sus propios requerimientos y eventos del registro.

Estructura general de los controladores ordinarios:

- comprobar los parámetros de entrada para ver si son válidos.
- comprobar si el dispositivo se encuentra en uso.
- Controlar el dispositivo, es decir, enviarle una secuencia de comandos
- comprobar si el controlador aceptó el comando y está preparado para aceptar el siguiente
- Una vez que se han emitido los comandos, se dará una de dos situaciones:
  - **El controlador de dispositivos debe esperar** hasta que el controlador realice cierto trabajo para él, **por lo que se bloquea** a sí mismo hasta que llegue la interrupción para desbloquearlo.
  - La operación termina sin retraso, por lo que el controlador no necesita bloquearse.
- Comprobar si hay errores.

- ❖ Pasar datos al software independiente del dispositivo
- ❖ Devolver información de estado para reportar los errores de vuelta al que lo llamó.
- ❖ Si hay otras peticiones en la cola se iniciara la siguiente y si no se bloqueara

Un controlador tiene que ser **reentrante**, es decir, un controlador en ejecución tiene que esperar a ser llamado una segunda vez antes de que se haya completado la primera llamada.

Software de E/S independiente del dispositivo

La función básica del software independiente del dispositivo es realizar las funciones de E/S que son comunes para todos los dispositivos y proveer una interfaz uniforme para el software a nivel de usuario.

Interfaz uniforme para controladores de dispositivos
Uso de búfer
Reporte de errores
Asignar y liberar dispositivos dedicados
Proporcionar un tamaño de bloque independiente del dispositivo

Figura 5-13. Funciones del software de E/S independiente del dispositivo.

Interfaz uniforme para controladores de dispositivos

Una cuestión importante en un sistema operativo es cómo hacer que todos los dispositivos de E/S y sus controladores se vean más o menos iguales. No es conveniente tener que modificar el sistema operativo para cada nuevo dispositivo.

Para ello se crea una **interfaz estandarizada** para cada controlador de dispositivo

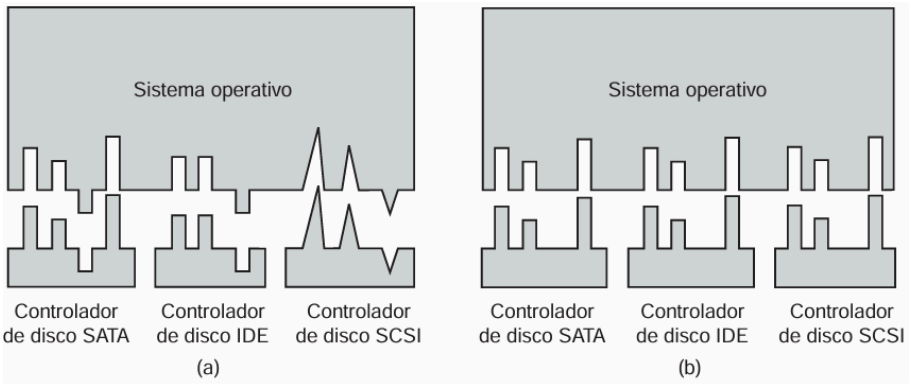


Figura 5-14. (a) Sin una interfaz de controlador estándar. (b) Con una interfaz de controlador estándar.

Para su funcionamiento, el S.O define un conjunto de funciones que el controlador debe proporcionar. Normalmente, el controlador contiene una **tabla con funciones propias** (Ej: Leer, encender, etc) y al ser cargado por el S.O se registra esta la dirección de esta tabla de apuntadores a funciones, permitiendo llamarlas. Esta tabla de apuntadores a funciones define la **interfaz entre el controlador y el resto del sistema operativo**.

Otro aspecto de tener una interfaz uniforme es la **forma de nombrar los dispositivos**. El software independiente del dispositivo se hace cargo de asignar nombres de dispositivo simbólicos al controlador apropiado.

Por ejemplo, en UNIX el nombre de un dispositivo especifica de manera única el nodo-i para un archivo especial, este nodo-i contiene el **número mayor de dispositivo**, el cual se utiliza para **localizar el controlador apropiado**, y el **número menor de dispositivo**, el cual se pasa como un parámetro al controlador para **poder especificar la unidad que se va a leer o escribir**.

ASPECTOS DE DISEÑO DEL SISTEMA OPERATIVO

OBJETIVOS DE DISEÑO

Hay 2 objetivos de suma importancia en el diseño del sistema de E/S:

- **Eficiencia**: Es importante debido a que las operaciones de E/S usualmente significan un cuello de botella en un computador, ya que **dispositivos de E/S son lentos** comparados a CPU o memoria principal.
  Para afrontar este problema se usa la **multiprogramación**, que permite que algunos procesos esperen por la finalización de operaciones de E/S mientras se está ejecutando otro proceso. También, se puede utilizar el **intercambio** para poder tener más procesos listos y mantener ocupado el procesador, aunque esto en sí es una operación de E/S.
- **Generalidad**: En busca de la **simplicidad y eliminación de errores**, es deseable manejar todos los **dispositivos de una manera uniforme**, aunque debido a la diversidad de estos dispositivos, es difícil. Para ello se utiliza una **estrategia modular jerárquica** para diseñar las funciones de E/S, esta estrategia, esconde la mayoría de los detalles del dispositivo de E/S en las rutinas de nivel inferior de manera que los procesos de usuario y los niveles más altos del sistema operativo **contemplan los dispositivos en términos de funciones generales** (lectura, escritura, abrir, etc).

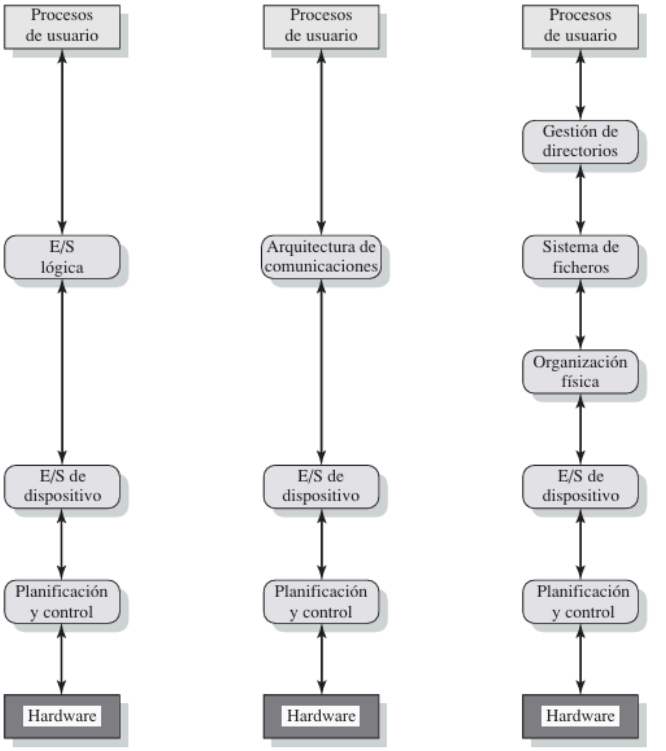
ESTRUCTURA LÓGICA DEL SISTEMA DE E/S

La filosofía de la naturaleza jerárquica de los sistemas operativos modernos se basa en que las **funciones del S.O** deberían estar **separadas en niveles** o capas de acuerdo a su **complejidad**, **escala de tiempo**, **nivel de abstracción**, etc.

Cada nivel realiza un subconjunto relacionado de las **funciones** requeridas del sistema operativo y **se apoya en el nivel inferior subyacente** para realizar funciones más básicas y ocultar los detalles de esas funciones, **proporcionando servicios al siguiente nivel superior**.
En general, los niveles inferiores tratan con una escala de tiempo mucho más corta (están más cerca del hardware), mientras que en el otro extremo, están las partes del S.O que se comunican con el usuario.

Considerando un dispositivo periférico local que se comunica de una manera sencilla, los niveles involucrados son los siguientes:

- **E/S lógica**: El módulo de E/S lógica trata a los dispositivos como un **recurso lógico** y **no se ocupa de los detalles del control real del dispositivo**, además se ocupa de la gestión de las tareas generales de E/S para los procesos de usuario, permitiendoles tratar con el dispositivo con mandatos sencillos (abrir, cerrar, etc).
- **E/S de dispositivo**: Las operaciones requeridas y los datos se convierten en las **secuencias apropiadas de instrucciones de E/S**, mandatos del canal y órdenes del controlador.
- **Planificación y control**: La gestión real de la cola y la planificación de las operaciones de E/S se producen en este nivel, así como el control de las operaciones. Por lo tanto, **en este nivel se manejan las interrupciones** y se recoge el estado de la E/S y se informa del mismo. Este es el **nivel de software que realmente interactúa con el módulo de E/S**, es decir, con el hardware del dispositivo.



(a) Dispositivo periférico local (b) Puerto de comunicación (c) Sistema de fichero

UTILIZACIÓN DE BUFFERS DE E/S

Supóngase que un proceso de usuario desea leer bloques de datos de una cinta de uno en uno, teniendo cada bloque una longitud de 512 bytes. La manera más sencilla sería enviar un mandato de E/S a la unidad de cinta y esperar hasta que los datos estén disponibles. Esta espera puede ser activa, o suspendiendo el proceso hasta que se produzca la interrupción.

Esta estrategia conlleva 2 problemas.

- El programa se queda esperando a que se complete la relativamente lenta operación de E/S.
- Esta estrategia de E/S **interfiere con las decisiones de intercambio del S.O**.

Para **solucionar** estos problemas, se puede utilizar la técnica de **E/S con buffers**, consiste en realizar las transferencias de entrada antes de que se hagan las peticiones correspondientes y llevar a cabo las transferencias de salida un cierto tiempo después de que haya hecho la petición.
Es importante en la **gestión de buffers**, distinguir los dispositivos de bloques y de carácter, ya que en los dispositivos de bloque es posible hacer referencia a los datos mediante su número de bloque, los cuales son los que se transfieren, mientras que los dispositivos de flujo de caracteres transfieren datos, tanto de entrada como de salida, como un flujo de caracteres.

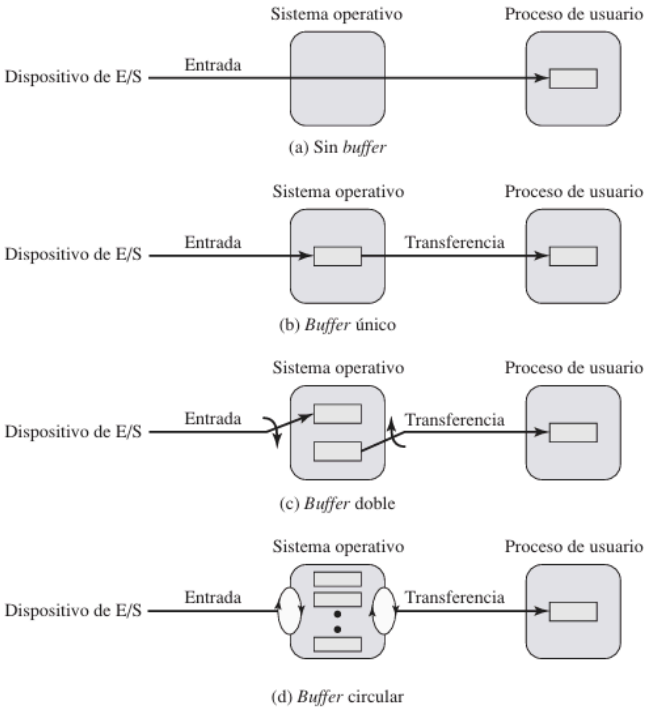


Figura 11.5. Esquemas de uso de *buffers* de E/S (entrada/salida).

Sin Buffer

Los datos se transfieren directamente entre el dispositivo de E/S y la memoria principal del sistema, sin almacenamiento temporal adicional.

Buffer Único

Cuando un proceso de usuario emite una petición de E/S, el sistema operativo asigna un buffer para la operación en la parte del sistema de la memoria principal.
El sistema operativo debe hacer un seguimiento de la asignación de buffers del sistema a los procesos de usuario.

Para **dispositivos orientados a bloques**, las transferencias de entrada usan el buffer del sistema, al completarse la transferencia, el proceso mueve el bloque al espacio de usuario e inmediatamente pide otro bloque, esto es denominado **lectura adelantada** o **entrada anticipada**.

En el caso de la **E/S orientada a flujo de caracteres**, el esquema de buffer único se puede utilizar en un modo de operación línea a línea o en uno byte a byte.

Esta **mejora la eficiencia** al permitir que la CPU continúe con otras tareas mientras los datos se transfieren entre el buffer y el dispositivo de E/S. Pero puede haber **cuellos de botella** si la velocidad de E/S no coincide con la velocidad de la CPU, ya que el buffer único puede llenarse o vaciarse rápidamente

Buffer Doble o Intercambio de Buffers

Es una mejora sobre la técnica del buffer único. En esta técnica, un proceso transfiere datos a (desde) un buffer mientras el sistema operativo vacía (o llena) el otro.

Utiliza dos buffers alternativos. Mientras uno de los buffers está siendo utilizado para la transferencia de datos, el otro puede ser procesado por la CPU o estar disponible para la siguiente operación de E/S. Después de completar una operación de E/S, los roles de los buffers se intercambian.

La **ventaja** es que **minimiza el tiempo de inactividad de la CPU** al proporcionar un **almacenamiento temporal más eficiente** y una **gestión más fluida de las operaciones de E/S**. Este debería suavizar el flujo de datos entre un dispositivo de E/S y un proceso. El buffer doble puede ser **inadecuado si el proceso realiza ráfagas rápidas de E/S**.

Buffer Circular

Si el interés está centrado en el rendimiento de un determinado proceso, donde se realizan rafagas rápidas de E/S, el problema puede solucionarse utilizando más de 2 buffers. Cuando se utilizan más de 2 buffer, al conjunto de estos buffers se lo denomina como **buffer circular**, siendo cada buffer individual una unidad del buffer circular.

Los datos se pueden transferir en múltiples etapas simultáneamente, con cada etapa utilizando un buffer diferente en el ciclo circular. Esto permite una gestión aún más eficiente y una reducción potencial de la latencia. La ventaja es que **optimiza el uso de memoria** al permitir una gestión continua de datos en movimiento y **reduce el riesgo de bloqueos** debido a esperas de E/S.

La Utilidad del Uso de Buffers

El uso de buffers es una **técnica que amortigua los picos en la demanda de E/S**. Aunque, por muchos buffers que se utilicen, estos no permitirán a un dispositivo de E/S mantener el ritmo de un proceso. Sin embargo, en un entorno de **multiprogramación**, donde hay diversas actividades de E/S y distintos procesos que hay que atender, el uso de buffers es una técnica que puede **incrementar la eficiencia del sistema operativo** y el rendimiento de los procesos individuales.

PLANIFICACIÓN DEL DISCO

Durante los últimos años, el incremento de la velocidad de los procesadores y de la memoria principal ha sobrepasado en gran medida la velocidad de acceso al disco

PARÁMETROS DE RENDIMIENTO DEL DISCO

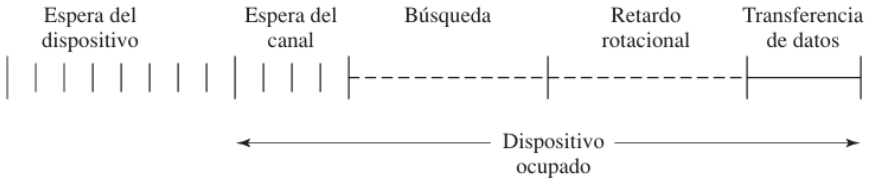


Figura 11.6. Diagrama de tiempos de una transferencia de E/S de disco.

Cuando está en funcionamiento la unidad de disco, el disco rota a una velocidad constante. Para leer o escribir, la cabeza se debe posicionar en la pista deseada y en el principio del sector requerido de dicha pista.

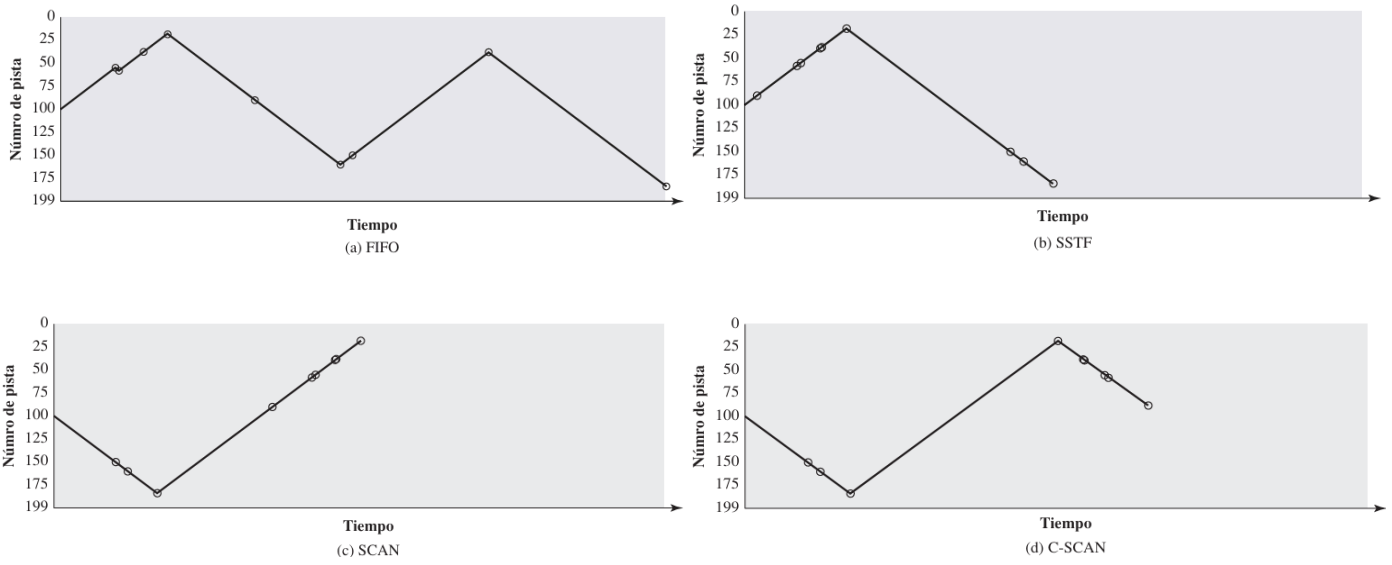
- **Tiempo de búsqueda**: El tiempo que se tarda en situar la cabeza en la pista
- **Retardo rotacional**: El tiempo que tarda en llegar el comienzo del sector hasta debajo de la cabeza

- **Tiempo de acceso**: Tiempo que se tarda en llegar a estar en posición para realizar la lectura o escritura (suma del tiempo de búsqueda y el retardo rotacional).
- **Tiempo de transferencia**: Tiempo requerido para la transferencia de datos de la operación (Se realiza lectura o escritura)

POLÍTICAS DE PLANIFICACIÓN DEL DISCO

La diferencia en el rendimiento se debe al tiempo de búsqueda. Si las peticiones de acceso a los sectores involucran una selección aleatoria de pistas, el rendimiento del sistema de E/S del disco será el peor posible. Para mejorar el asunto, se necesita reducir el tiempo medio gastado en las búsquedas, y para ello, utilizamos diferentes algoritmos.

(a) FIFO (comenzando en la pista 100)		(b) SSTF (comenzando en la pista 100)		(c) SCAN (comenzando en la pista 100, en la dirección de números de pista crecientes)		(d) C-SCAN (comenzando en la pista 100, en la dirección de números de pista crecientes)	
Próxima pista accedida	Número de pistas atravesadas	Próxima pista accedida	Número de pistas atravesadas	Próxima pista accedida	Número de pistas atravesadas	Próxima pista accedida	Número de pistas atravesadas
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
longitud media de búsqueda	55,3	longitud media de búsqueda	27,5	longitud media de búsqueda	27,8	longitud media de búsqueda	35,8



Primero en entrar, primero en salir (FIFO)

La forma más sencilla de planificación corresponde con el algoritmo del primero en entrar (FIFO), este **procesa los elementos de la cola en orden secuencial**. Tiene la **ventaja** de ser **equitativo.**
Sin embargo, con muchos procesos compitiendo por el disco será igual en cuanto a rendimiento que la planificación aleatoria.

Primero el de tiempo de servicio más corto (SSTF)

La política **SSTF** consiste en **seleccionar la petición de E/S del disco que requiera un menor movimiento del brazo desde su posición actual**. De este modo, siempre se realiza una selección de manera que se produzca un tiempo de búsqueda mínimo, aunque no garantiza obtener el menor tiempo de búsqueda promedio. Este algoritmo, debería proporcionar un **mejor rendimiento que FIFO**. Dado que el brazo puede moverse en dos direcciones, en caso de igualdad de distancia, se utilizaria un algoritmo aleatorio para decidir dónde ir.

Scan

Con la excepción de la planificación FIFO, las demás políticas o esquemas pueden dejar alguna petición sin servir hasta que se vacíe completamente la cola de peticiones. Una alternativa que impide esto es el algoritmo SCAN (ascensor).
En este esquema, **el brazo sólo debe moverse en una dirección**, satisfaciendo todas las peticiones pendientes que encuentre en su camino, **hasta que alcanza la última pista en esa dirección** (se lo denomina LOOK). En ese momento, la dirección de servicio del brazo se invierte y la búsqueda continúa en la dirección opuesta, sirviendo de nuevo todas las peticiones en orden.
La política SCAN **favorece** a los trabajos cuyas peticiones corresponden con las pistas más cercanas tanto a las pistas interiores como a las exteriores, así como a los trabajos que han llegado más recientemente.

C-Scan

La política C-SCAN (SCAN circular) **restringe la búsqueda a una sola dirección**. Por tanto, después de visitar la última pista en una dirección, el brazo vuelve al extremo opuesto del disco y la búsqueda comienza de nuevo. Esto **reduce el retardo máximo que pueden experimentar las nuevas peticiones**.

Raid - Vector Redundante de Discos Independientes

Si un determinado componente ya no puede mejorarse más, para lograr mejoras adicionales en el rendimiento se van a tener que utilizar múltiples componentes en paralelo. En el caso de los discos, usando múltiples discos, se pueden manejar en paralelo las peticiones de E/S independientes, siempre que los datos solicitados residan en distintos discos.

El **Vector Redundante de Discos Independientes** (**RAID**), es un **esquema estándar para el diseño de bases de datos en múltiples discos**, este consta de 7 niveles (Del 0 al 6) y no implican una relación jerárquica, sino que designan a diversas arquitecturas de diseño que comparten **3 características comune**s:

1. RAID corresponde con un **conjunto de unidades físicas de disco tratadas** por el sistema operativo **como un único dispositivo lógico**.
2. Los **datos** están **distribuidos a lo largo de las unidades físicas de un vector**.
3. La **capacidad de redundancia** del disco se utiliza para almacenar información de paridad, que **garantiza que los datos se pueden recuperar** en caso de que falle un disco.

Esta nueva estrategia reemplazó los dispositivos de disco de gran capacidad por múltiples dispositivos de menor capacidad distribuyendo los datos de manera que se permitan los accesos simultáneos a los datos de múltiples dispositivos. Con ello, **se mejora el rendimiento de E/S** y **se facilita el crecimiento gradual en la capacidad**.

Tabla 11.4. Niveles RAID.

Categoria	Nivel	Descripción	Discos implicados	Disponibilidad de datos	Capacidad de transferencia para datos de E/S grandes	Tasa para peticiones de E/S pequeñas
En bandas	0	Sin redundancia	N	Inferior a un único disco	Muy alta	Muy alta tanto para lecturas como para escrituras
Espejo	1	Discos duplicados	$2N$, $3N$, etc.	Mayor que RAID 2, 3, 4 o 5; menor que RAID 6	Mayor que un único disco para lecturas; similar a un único disco para escrituras	Hasta el doble de un único disco para lecturas; similar a un único disco para escrituras
Acceso paralelo	2	Redundancia mediante Código Hamming	$N + m$	Mucho mayor que un único disco; mayor que RAID 3, 4 o 5	La mayor de todas las alternativas mostradas	Aproximadamente el doble de un único disco
	3	Paridad intercalada a nivel de bit	$N + 1$	Mucho mayor que un único disco; comparable a RAID 2, 4 ó 5	La mayor de todas las alternativas mostradas	Aproximadamente el doble de un único disco
Acceso independiente	4	Paridad intercalada a nivel de bloque	$N + 1$	Mucho mayor que un único disco; comparable a RAID 2, 3 o 5	Similar a RAID 0 para lecturas; significativamente inferior a un único disco para escrituras	Similar a RAID 0 para lecturas; significativamente inferior a un único disco para escrituras
	5	Paridad distribuida e intercalada a nivel de bloque	$N + 1$	Mucho mayor que un único disco; comparable a RAID 2, 3 o 4	Similar a RAID 0 para lecturas; inferior a un único disco para escrituras	Similar a RAID 0 para lecturas; generalmente inferior a un único disco para escrituras
	6	Paridad dual distribuida e intercalada a nivel de bloque	$N + 2$	La mayor de todas las alternativas mostradas	Similar a RAID 0 para lecturas; inferior a RAID 5 para escrituras	Similar a RAID 0 para lecturas; significativamente inferior a RAID 5 para escrituras

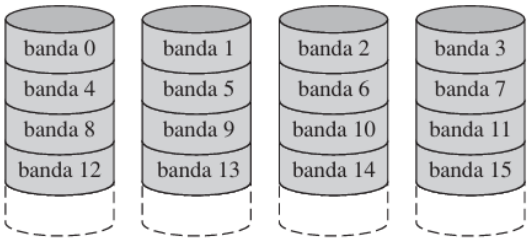
RAID de Nivel 0

RAID de nivel 0 no es un verdadero miembro de la familia RAID, puesto que no incluye redundancia para mejorar la fiabilidad.
En RAID 0, **los datos de los usuarios y del sistema están distribuidos a lo largo de todos los discos del vector**. Estos datos están distribuidos en **bandas** (strips) a lo largo de los discos disponibles. La gran **ventaja** de esto, radica en que si están pendientes **dos peticiones de E/S diferentes** que solicitan dos bloques de datos distintos, **hay una gran probabilidad de que los bloques pedidos estén en diferentes discos.** Por tanto, se pueden llevar a cabo las dos peticiones en paralelo, **reduciendo el tiempo de espera en la cola de E/S.**

RAID 0 para una elevada capacidad de transferencia de datos
Para que las aplicaciones experimenten esa alta tasa de transferencia, se deben cumplir dos requisitos:

1. Debe existir una elevada capacidad de transferencia a lo largo de todo el camino entre la memoria de la máquina y las unidades de disco individuales.
2. La aplicación debe hacer peticiones de E/S que accedan eficientemente al vector de discos.

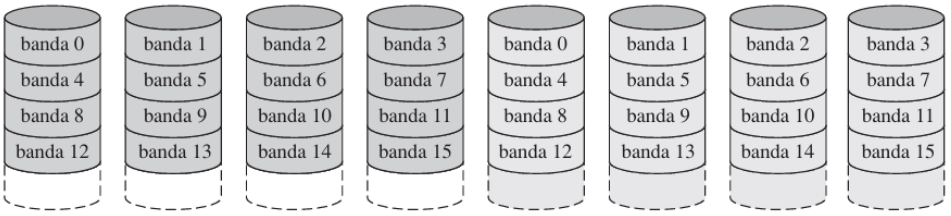
RAID 0 para una elevada tasa de peticiones de E/S
En un entorno de transacciones, habrá cientos de peticiones de E/S por segundo. Un vector de discos puede proporcionar tasas elevadas de ejecución de E/S repartiendo la carga de E/S entre los múltiples discos. El reparto de carga efectivo sólo se alcanza si normalmente hay múltiples peticiones de E/S pendientes.



(a) RAID 0 (sin redundancia)

RAID de Nivel 1

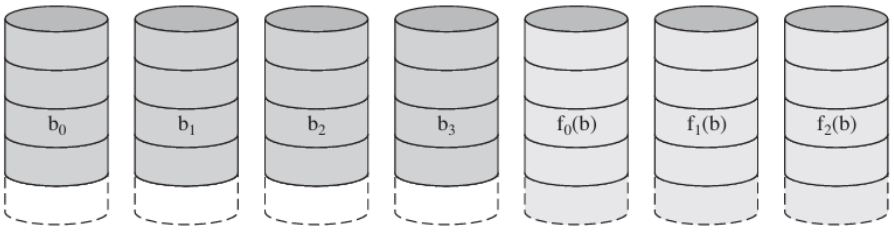
El nivel RAID 1 se diferencia de los niveles del 2 al 6 en el modo en que se logra la redundancia. En RAID 1, se consigue la redundancia mediante la **duplicación de todos los datos**. En esta estrategia, se proporciona una copia de respaldo en tiempo real de todos los datos.
La principal **desventaja** de RAID 1 es el **coste**: **requiere el doble de espacio de disco** que el correspondiente al disco lógico proporcionado.



(b) RAID 1 (discos espejo)

RAID de Nivel 2

Los niveles RAID 2 y 3 utilizan una **técnica de acceso paralelo**. En esta técnica, todos los miembros del disco participan en la ejecución de cada petición de E/S. Normalmente, los ejes de las distintas **unidades se sincronizan** de manera que en todo momento **la cabeza de cada disco esté en la misma posición en todos los discos**.
En RAID 2, también se utiliza una distribución de datos en **bandas**, las cuales son **muy pequeñas**.
Con RAID 2, se calcula un código de **corrección de error** **con los bits correspondientes de cada disco de datos**, almacenándose los bits del código resultante en las correspondientes posiciones de bit en los múltiples discos de paridad, utilizando un Código Hamming. La **desventaja** es que sigue siendo **bastante costoso**.

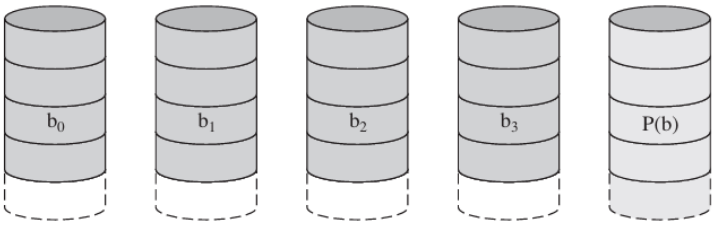


(c) RAID 2 (redundancia mediante código Hamming)

RAID de Nivel 3

El esquema RAID 3 se organiza de una manera similar al usado en RAID 2. La diferencia es que RAID 3 requiere sólo un disco redundante, con independencia del tamaño del vector de discos.
RAID 3 emplea **acceso paralelo**, teniendo los **datos distribuidos en pequeñas bandas**. **En lugar de un código de corrección de errores, se calcula un bit de paridad** simple para el conjunto de bits almacenados en la misma posición en todos los discos de datos.

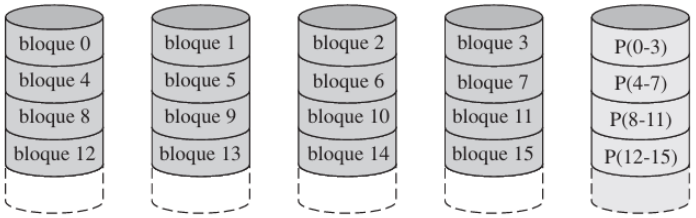
Redundancia: En el caso de que ocurra un fallo en un dispositivo, se accede al dispositivo de pari dad y se reconstruyen los datos desde los dispositivos restantes
Rendimiento: Dado que los datos están distribuidos en pequeñas bandas, el esquema RAID 3 puede alcanzar velocidades de transferencia de datos muy elevadas.



(d) RAID 3 (paridad a nivel de bit)

RAID de Nivel 4

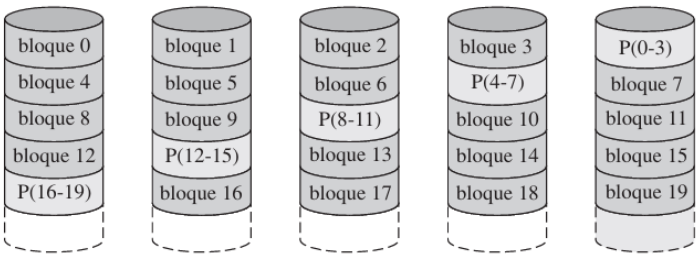
Los niveles RAID de 4 a 6 utilizan una técnica de **acceso independiente**. En estos vectores de acceso independiente, **cada disco del vector opera independientemente**, de manera que se pueden servir en paralelo peticiones de E/S independientes. Siendo más **efectivos para las aplicaciones que requieren tasas elevadas de peticiones de E/S** y **menos adecuados para aquéllas que necesitan tasas elevadas de transferencias de datos**.
Como en los otros esquemas RAID, se utiliza una distribución de datos en bandas, aunque en los esquemas RAID desde el 4 al 6, **las bandas son relativamente grandes**.



(e) RAID 4 (paridad distribuida a nivel de bloque)

RAID de Nivel 5

El esquema RAID 5 se organiza de manera similar al RAID 4. La diferencia es que el esquema RAID 5 **distribuye las bandas de paridad a través de todos los discos**. La asignación habitual usa un esquema rotatorio.
La distribución de las bandas de paridad a través de todos los dispositivos evita el potencial cuello de botella de E/S debido a la existencia de un único disco de paridad que aparece en el esquema RAID 4.

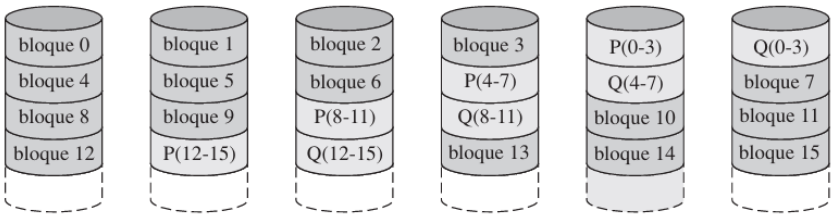


(f) RAID 5 (paridad distribuida a nivel de bloque)

RAID de Nivel 6

En el esquema RAID 6, se realizan dos cálculos de paridad diferentes, almacenandose en bloques separados de distintos discos. La **ventaja** del esquema RAID 6 es que proporciona una extremadamente alta disponibilidad de datos.

Por otro lado, el esquema RAID 6 incurre en una penalización de escritura sustancial, debido a que cada escritura afecta a dos bloques de paridad.



(g) RAID 6 (redundancia dual)

CACHÉ DE DISCO

Memoria cache: Memoria que es más pequeña y más rápida que la memoria principal y que se interpone entre la memoria principal y el procesador.
Caché de disco: **Es un buffer en memoria principal para almacenar sectores del disco**. Mismo principio que la memoria caché, aplicada a la memoria del disco.

CONSIDERACIONES DE DISEÑO

Hay varios aspectos de diseño que son de interés:

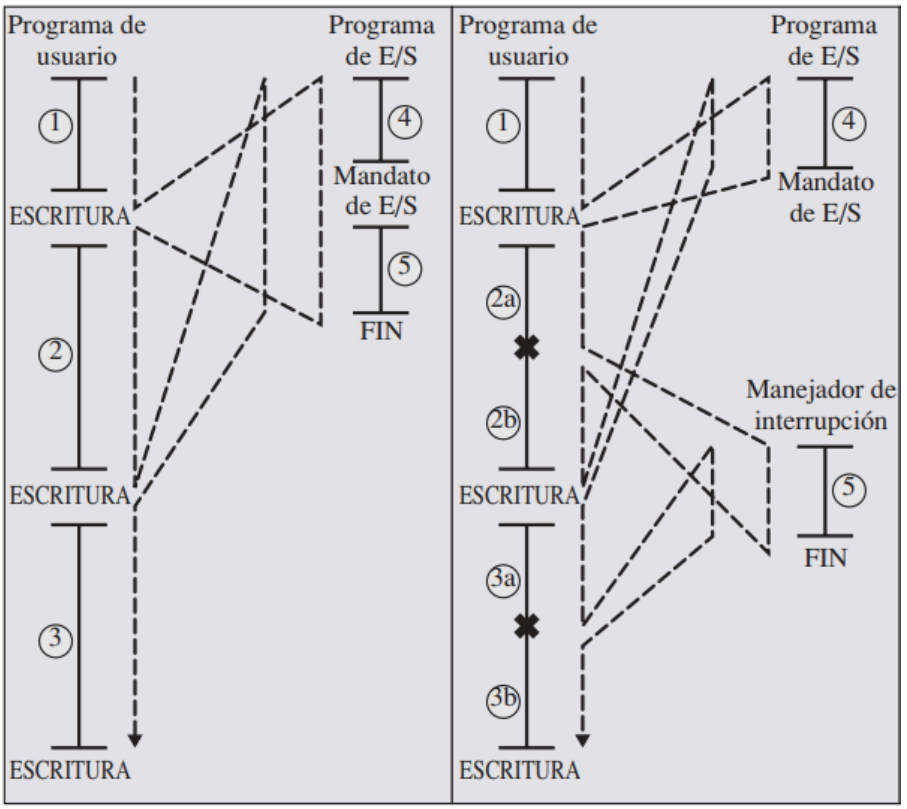
- En primer lugar, cuando se satisface una petición de E/S de la caché de disco, se deben **entregar los datos de la caché al proceso solicitante**. Esta operación se puede hacer:
  - Copiando el bloque de datos almacenado en la memoria principal asignada a la caché de disco hasta la memoria asignada al proceso de usuario.
  - Utilizando simplemente la técnica de la memoria compartida.
- Un segundo aspecto de diseño está relacionado con la **estrategia de reemplazo**. Cuando se trae un nuevo sector a la caché de disco, se debe reemplazar uno de los bloques existentes. Para ello, se han probado diversos algoritmos:
  - **Menos recientemente usado (LRU)**: Se reemplaza el bloque que ha estado en la caché más tiempo sin ser accedido. Es el algoritmo más frecuentemente utilizado.
  - **Menos frecuentemente usado (LFU):** Se reemplaza el bloque del conjunto que ha experimentado la menor cantidad de referencias.

Interrupciones

Interrupciones: Es un **mecanismo** por el cual otros módulos (memoria y E/S) pueden **interrumpir** el **secuenciamiento normal del procesador**. Constituyen una manera de mejorar la utilización del procesador.

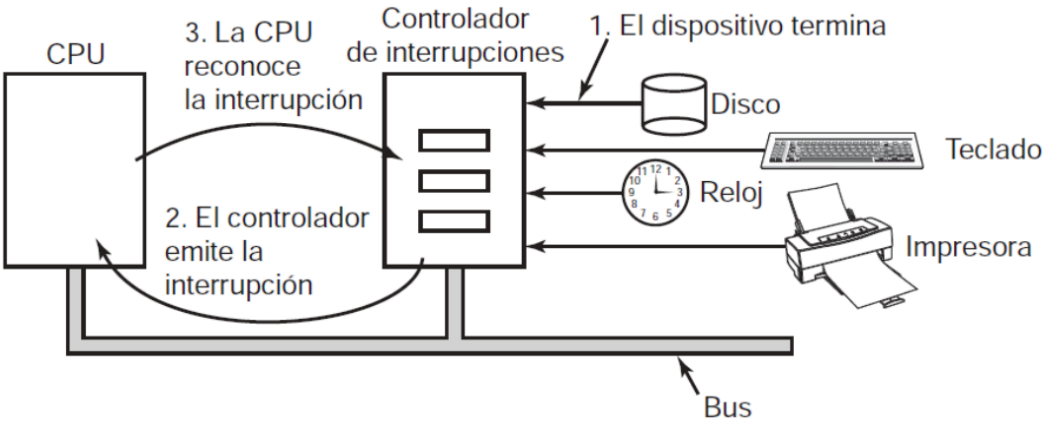
Tabla 1.1. Clases de interrupciones.

De programa	Generada por alguna condición que se produce como resultado de la ejecución de una instrucción, tales como un desbordamiento aritmético, una división por cero, un intento de ejecutar una instrucción de máquina ilegal, y las referencias fuera del espacio de la memoria permitido para un usuario.
Por temporizador	Generada por un temporizador del procesador. Permite al sistema operativo realizar ciertas funciones de forma regular.
De E/S	Generada por un controlador de E/S para señalar la conclusión normal de una operación o para indicar diversas condiciones de error.
Por fallo del hardware	Generada por un fallo, como un fallo en el suministro de energía o un error de paridad en la memoria.



(a) Sin interrupciones (b) Interrupciones; espera de E/S breve

Tratamiento de Interrupciones



Cuando un dispositivo de E/S termina el trabajo asignado, produce una interrupción. Para ello, impone una señal en una línea de bus que se le haya asignado. Esta señal es detectada por el chip controlador de interrupciones en la tarjeta principal, que finalmente decide qué hacer con ella.

Interrupciones y el Ciclo de Instrucción

Gracias a las interrupciones, el **procesador** puede dedicarse a **ejecutar otras instrucciones mientras que la operación de E/S se está llevando a cabo**.

¿Cómo funciona?

Cuando el **dispositivo externo está listo** para ser atendido (preparado para aceptar más datos del CPU), el **módulo de E/S de este dispositivo** externo **manda una señal** de **petición de interrupción** al procesador. El **procesador responde suspendiendo la ejecución del programa actual**, saltando a la **rutina de servicio específica de este dispositivo de E/S**, conocida como **manejador de interrupción**, y reanudando la ejecución original después de haber atendido al dispositivo.
Estas **interrupciones** se pueden **producir en cualquier punto de la ejecución del programa**, no solo en una determinada instrucción.

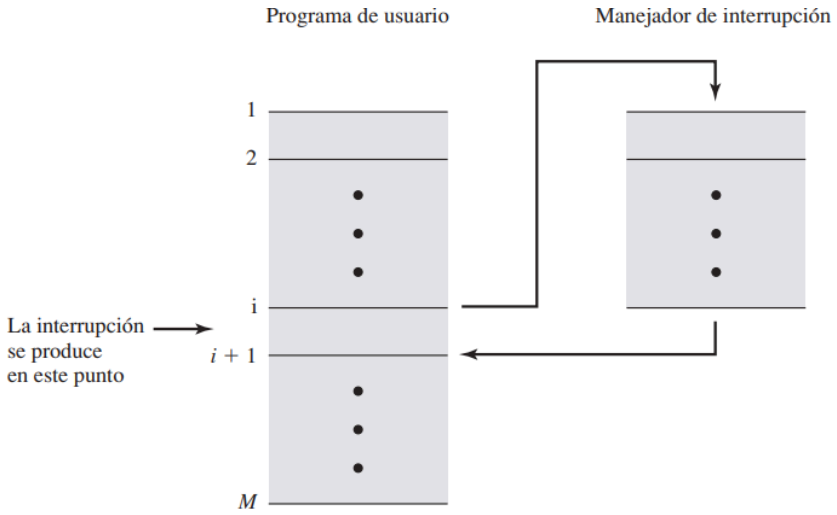


Figura 1.6. Transferencia de control mediante interrupciones.

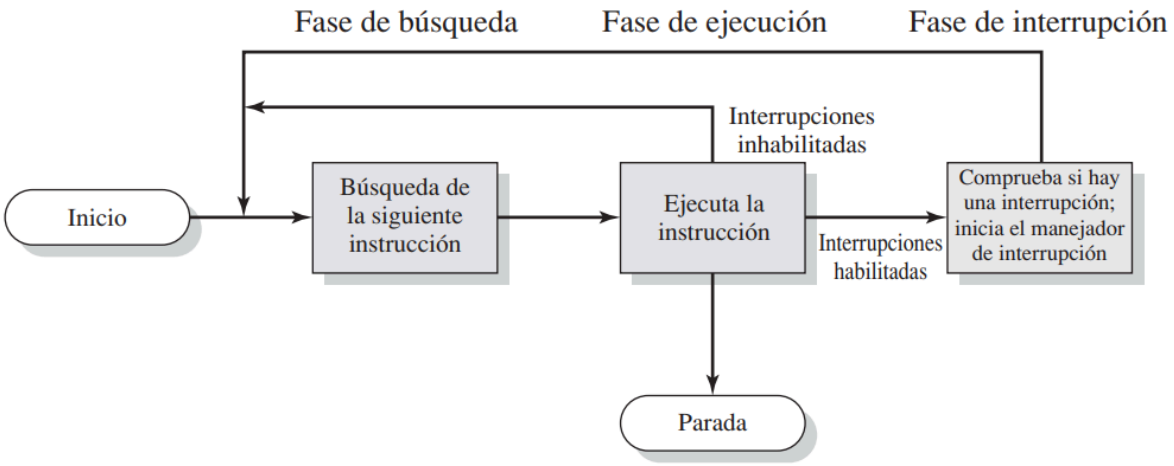


Figura 1.7. Ciclo de instrucción con interrupciones.

Para tratar las interrupciones, se añade una **fase de interrupción** al ciclo de instrucción, en esta fase, el procesador comprueba si se ha producido cualquier interrupción, hecho indicado por la presencia de una señal de interrupción:

- **Si no hay interrupciones pendientes**: El procesador continúa con la fase de búsqueda y lee la siguiente instrucción del programa actual.
- **Si hay interrupciones pendientes**: El **procesador suspende la ejecución del programa** actual y **ejecuta la rutina del manejador de interrupción**. Esta rutina determina la naturaleza de la interrupción y realiza las acciones que se requieran y **forma parte del S.O**.

Procesamiento de Interrupciones

La aparición de una interrupción dispara varios eventos, tanto en el hardware del procesador como en el software.
Es importante **salvar** toda la información de **estado del programa interrumpido para su posterior reanudación**, ya que **la interrupción es imprevisible**, no es una rutina llamada desde el programa, si no que puede suceder en cualquier momento y en cualquier punto de la ejecución.

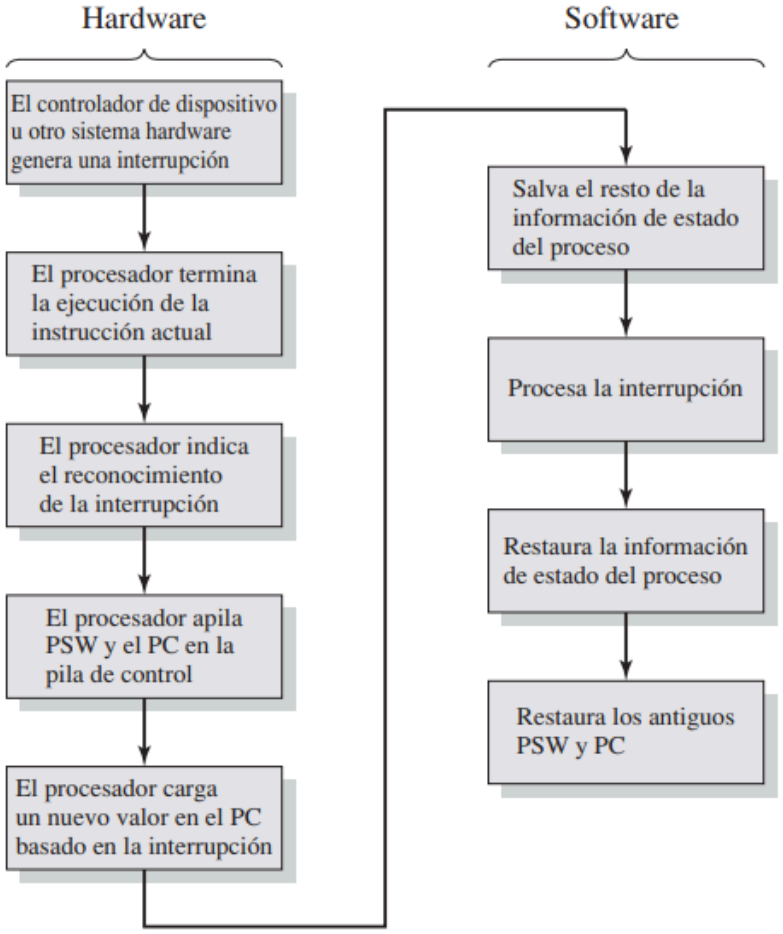


Figura 1.10. Procesamiento simple de interrupciones.

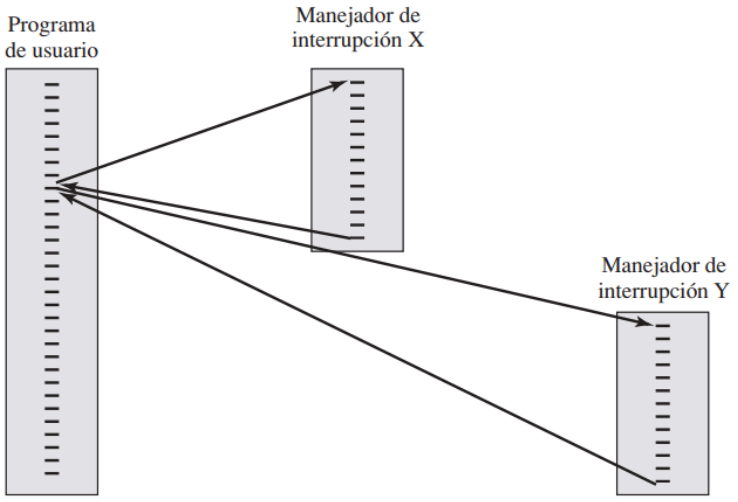
Múltiples Interrupciones

Se pueden considerar dos alternativas a la hora de tratar con múltiples interrupciones:

- **Interrupción Inhabilitada**: El procesador ignorará cualquier nueva señal de petición de interrupción. Si se produce una interrupción durante este tiempo, permanecerá pendiente de ser procesada, de manera que el procesador sólo la comprobará después de que se rehabiliten las interrupciones.

La **desventaja** es que **no tiene en cuenta la prioridad** relativa o el grado de urgencia de las

interrupciones.



(a) Procesamiento secuencial de interrupciones

- **Definir Prioridades**: Definir prioridades para las interrupciones y permitir que una interrupción de más prioridad cause que se interrumpa la ejecución de un manejador de una interrupción de menor prioridad.

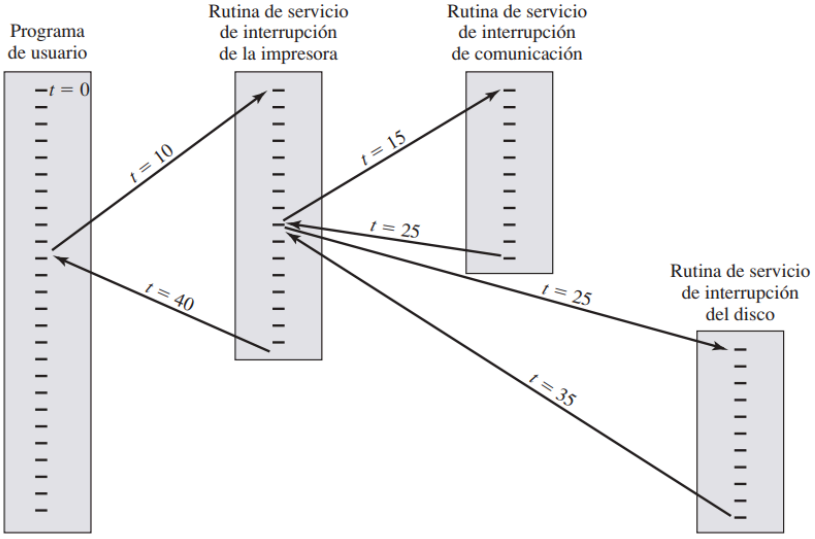


Figura 1.13. Ejemplo de secuencia de tiempo con múltiples interrupciones.

Multiprogramación

Una solución para poder utilizar eficientemente el procesador puede ser **permitir que múltiples programas de usuario estén activos al mismo tiempo**.

Sistemas Operativos Distribuidos y Tiempo Real

A partir de la década de 1980, 2 avances tecnológicos surgi
eron:

- Mejoras en los cómputos, es decir, mejores procesadores a un precio sumamente económico.
- **Invención de redes**:
  - **Redes de área local (LAN)**: Permiten conectar muchas máquinas dentro de un edificio de tal forma que se pueden transferir pequeñas cantidades de información entre ellas a gran velocidad.

- ➢ **Redes de área amplia (WAN)**: Permiten que millones de máquinas en toda la Tierra se conecten con velocidades que varían.

El resultado de estas tecnologías es que es posible reunir sistemas de cómputo compuestos por un gran número de CPU conectados mediante una red de alta velocidad. Estos reciben el nombre de sistemas distribuidos.

Sistema Distribuido: Colección de computadoras independientes que aparecen ante los usuarios del sistema como una única computadora.

Objetivos

Ventajas

Costo: Tienen en potencia una **proporción precio/desempeño** mucho mejor que la de un sistema centralizado.
Rendimiento: Puede producir mejor rendimiento del que podría dar cualquier mainframe a cualquier precio.

Los dos anteriores hacen referencia a la descentralización y un mayor nivel de cómputo.

Trabajo cooperativo apoyado por computadora: Diseñados para que muchos **usuarios trabajen en forma conjunta sin importar la distancia**.
Confiabilidad: Al distribuir la carga de trabajo en muchas máquinas, **la falla de un circuito afectará a una sola máquina y no al resto**.
Escalabilidad de hardware: Podrían **añadirse más procesadores al sistema** según sea necesario.
Compartir recursos: Los usuarios pueden compartir datos y periféricos caros, logrando una mejor comunicación entre ellos.
Mayor flexibilidad potencial: Darle a cada usuario una computadora personal aislada.

Desventajas

Falta de experiencia en el diseño, implantación y uso del software distribuido.
Dependencia a las redes de comunicación: La pérdida o saturación de la red puede negar algunas de las ventajas que el sistema distribuido debía conseguir.
Seguridad en el acceso a la información compartida: Si las personas pueden tener acceso a los datos en todo el sistema, entonces también pueden tener acceso a datos que no deberían ver.

Aspectos del Diseño

Analizaremos aspectos claves del diseño de un sistema operativo distribuido.

Transparencia

Consiste en la forma de lograr la **imagen de un único sistema**. En otras palabras, es como se consigue que las personas piensen que la colección de máquinas es tan sólo un sistema de tiempo compartido de un procesador
La transparencia se puede lograr en dos niveles distintos:

- ❖ Ocultar la distribución a los usuarios

- Transparente para los programas

Existen distintos **tipos de transparencias** en un sistema distribuido:

- **Transparencia de localización**: Se refiere a que los usuarios no pueden indicar la localización de los recursos de hardware y software (Ej: Como los CPU, impresoras, archivos, etc).
- **Transparencia de migración**: Significa que los recursos deben moverse de una posición a otra sin tener que cambiar sus nombres.
- **Transparencia de réplica**: El S.O es libre de fabricar por su cuenta copias adicionales de los archivos y otros recursos sin que lo noten los usuarios.
- **Transparencia con respecto al concurrismo**: Varios usuarios pueden compartir recursos de manera automática, incluso, si dos o más de ellos intentan acceder al mismo recurso al mismo tiempo.
- **Transparencia con respecto al paralelismo**: Los programadores que deseen utilizar varios CPU para un problema, deberán programarlo de forma explícita.

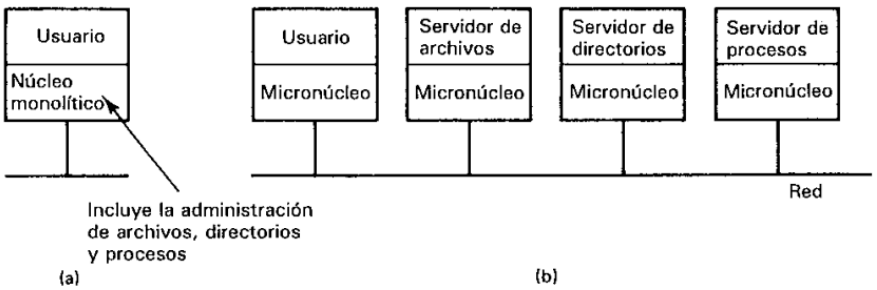
Tipo	Significado
Transparencia de localización	Los usuarios no pueden indicar la localización de los recursos
Transparencia de migración	Los recursos se pueden mover a voluntad sin cambiar sus nombres
Transparencia de réplica	Los usuarios no pueden indicar el número de copias existentes
Transparencia de concurrencia	Varios usuarios pueden compartir recursos de manera automática
Transparencia de paralelismo	Las actividades pueden ocurrir en paralelo sin el conocimiento de los usuarios

Flexibilidad

Puesto que el campo se encuentra todavía en sus inicios, el diseño se debe hacer con la idea de facilitar los cambios futuros.

Existen **dos escuelas de pensamiento** en cuanto a la estructura de los sistemas distribuidos:

- **Núcleo Monolítico**: Cada máquina debe ejecutar un núcleo tradicional que proporcione la mayoría de los servicios. Su única **ventaja** es el rendimiento
- **Núcleo Microkernel**: El núcleo debe proporcionar lo menos posible y que el grueso de los servicios del sistema operativo se obtenga a partir de los servidores a nivel usuario. La **ventaja** de este método es que es **altamente modular** (existe una interfaz bien definida con cada servicio), esta capacidad de añadir, eliminar y modificar servicios es lo que le da su **flexibilidad**. Además, **los usuarios tienen la libertad de escribir sus propios servicios**.



Confiabilidad

Uno de los **objetivos** originales de los sistemas distribuidos fue hacerlos **más confiables que los sistemas de un procesador**. La idea es que **si una máquina falla**, alguna **otra máquina se encargue del trabajo**.

Es importante distinguir entre varios aspectos de la confiabilidad:

- **Disponibilidad**: Se refiere a la fracción de tiempo en que se puede utilizar el sistema. Un sistema confiable debe ser muy disponible.
- **Información Consistente**: Los datos confiados al sistema no deben perderse o mezclarse de manera alguna.
- **Seguridad**: Los archivos y otros recursos deben ser protegidos contra el uso no autorizado.
- **Tolerancia de Fallas**:Se deben diseñar de forma que escondan las fallas, es decir, ocultarlos de los usuarios.

Desempeño

Un sistema distribuido debe tener buen desempeño, ya que por más transparente, flexible y confiable que sea, si no posee un buen desempeño no será utilizable.
Se pueden utilizar diversas métricas del desempeño:

- Tiempo de respuesta
- Rendimiento (Número de trabajos por hora)
- Uso del sistema
- Cantidad consumida de la capacidad de red

El **problema** del desempeño es que la **comunicación** (Envío de un mensaje y respuesta en una LAN) **es lenta**. Esto **se debe** a un manejo inevitable del **protocolo** en ambos extremos.
Entonces, **para optimizar el desempeño**, hay que **minimizar el número de mensaje**s. La dificultad con esto es que para mejorar el desempeño, hay que tener muchas actividades en ejecución en paralelo, lo cual requiere el envío de muchos mensajes.

Una posible solución es prestar atención al **tamaño de grano** de todos los cálculos:

- **Paralelismo de grano fino**: Trabajos que implican gran número de pequeños cálculos y que interactúan en gran medida con otros. Estos pueden ser la causa de algunos problemas en los sistemas distribuidos con una comunicación lenta.
- **Paralelismo de grano grueso**: Trabajos que implican grandes cálculos y bajas tasas de interacción, así como pocos datos. Estos **se ajustan mejor** a los modelos de **sistemas distribuidos**.

Escalabilidad

Antes de que pase mucho tiempo, tendremos un sistema distribuido con decenas de millones de usuarios. La cuestión es si los métodos que se desarrollan en la actualidad podrán escalar hacia tales grandes sistemas.
Existen distintos cuellos de botella que los diseñadores deben intentar evitar en los sistemas distribuidos de gran tamaño:

- **Componentes centralidad**. Un solo servidor de correo para todos los usuarios.
- **Tablas centralizadas**. Un directorio telefónico en línea.
- **Algoritmos centralizados**. Realización de un ruteo con base en la información completa.

Necesidad de una Arquitectura de Protocolos

Cuando un computador, terminal y/u otro dispositivo de procesamiento de datos intercambia datos, el procedimiento involucrado puede ser muy complejo.

Comunicación de computadoras: Intercambio de información entre computadores con finalidad cooperativa
Red de computadoras: Cuando uno o más computadores se interconectan a través de una red de comunicación.

En relación a la comunicación de computadores y redes de computadores, hay **2 conceptos** de suma importancia:

- **Protocolos**: Conjunto de reglas que gobiernan el intercambio de datos entre dos entidades (Ej: programas). Los **elementos principales** de un protocolo son:
  - **Sintaxis**: Incluye cosas tales como formatos de datos y niveles de señales
  - **Semántica**: Incluye información de control para realizar coordinación y gestión de errores
  - **Temporización**: Incluye ajuste de velocidades y secuenciamiento
- **Arquitectura de protocolos**: Conjunto estructurado de módulos que implementan la función de comunicaciones.

Interfaz de programación de aplicaciones (API): Un conjunto de funciones y programas que permiten a los clientes y servidores intercomunicarse.
Base de datos relacional: Una base de datos en la que el acceso a la información está restringido por la selección de filas que satisfacen todos los criterios de búsqueda.
Lenguaje Estructurado de Consultas (SQL): Lenguaje desarrollado por IBM y estandarizado por ANSI que permite acceder, crear, actualizar e interrogar bases de datos relacionales.

Computación Cliente/Servidor

La característica fundamental de una arquitectura cliente/servidor es la distribución de las tareas de la aplicación entre el cliente y el servidor.
Se trata de **3 elementos fundamentales**:

- **Cliente**: Un elemento de la red que solicita información, puede interrogar a una base de datos o solicitar información de un servidor. Normalmente simples PCs o estaciones de trabajo que proporcionan una interfaz de fácil manejo al usuario final (Ej: Windows).
- **Servidor**: Un computador, normalmente una estación de trabajo de gran potencia, un minicomputador, o un mainframe, que almacena la información para los clientes de la red. Proporciona un conjunto de servicios compartidos a los clientes (Ej: Una base de datos).
- **Red**: Usuarios, aplicaciones y recursos se unen a través de una LAN, WAN o Internet.

Tanto en el cliente como en el servidor, el software básico es el sistema operativo que ejecuta sobre el hardware de la plataforma. No importa qué sistema operativo posea cada uno de ellos (plataforma, cliente y servidor), siempre que los **clientes y los servidores compartan los mismos protocolos de comunicación** y soporten las mismas aplicaciones, estas diferencias de bajo nivel no son relevantes. Quien permite que interactúen el cliente y el servidor es el **software de comunicaciones** (Ej: TCP/IP).

El **objetivo** de todo este software de soporte (comunicaciones y sistema operativo) es **proporcionar las bases para las aplicaciones distribuidas**. Las **funciones** que realiza una aplicación **se pueden dividir entre el cliente y el servidor**, de manera que se optimice el uso de los recursos.

Sockets

Sockets: Permite la **comunicación entre un proceso cliente y un proceso servidor** y puede ser orientado a conexión o no orientado a conexión.
Un socket de cliente en un computador utiliza una dirección para llamar a un socket de servidor en otro computador. Una vez que han entrado en comunicación los sockets, los dos computadores pueden intercambiar datos.

La concatenación de un valor de puerto y una dirección IP forma un socket, que es único en Internet. **Se utiliza para definir una** interfaz de programación de aplicaciones (**API**), que es una interfaz genérica de comunicaciones para escribir programas que usan TCP y UDP.

En la práctica, un socket se identifica por un triplete: protocolo (TCP o UDP), dirección local (dirección de IP), proceso local (número de puerto). El API de Sockets reconoce tres tipos de sockets: sockets stream, sockets datagrama y sockets raw

Middleware

El desarrollo y utilización de los productos cliente/servidor ha sobrepasado con mucho los esfuerzos para estandarizar todos los aspectos de la computación distribuida.
Esta falta de estándares genera un **problema de interoperabilidad**, el cual viene a ser resuelto por el middleware.

Middleware: Interfaces de programas y protocolos estándares entre la aplicación y el software de comunicaciones y el S.O
El **propósito básico** del middleware es **permitir** a una aplicación o usuario en el **cliente acceder** a una variedad de **servicios en el servidor sin preocuparse de las diferencias entre los servidores**.

El middleware tiene la capacidad de **esconder la complejidad y disparidad** de los diferentes protocolos de red y sistemas operativos. Proporciona una capa de software que **permite un acceso uniforme** a estos sistemas diferentes.
El middleware, que se sitúa entre todas las plataformas cliente y servidor, **es el responsable de guiar las peticiones al servidor apropiado**.

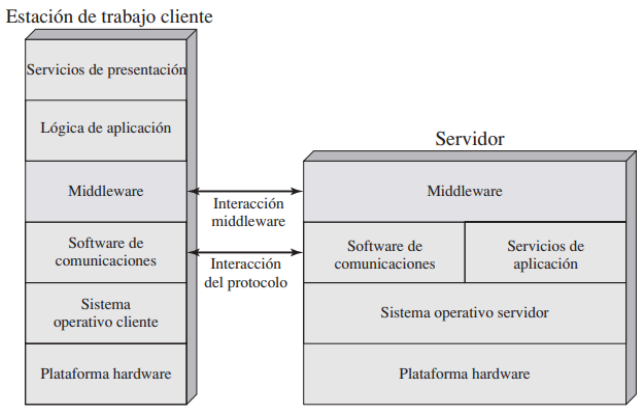


Figura 14.8. El papel del middleware en la arquitectura cliente/servidor.

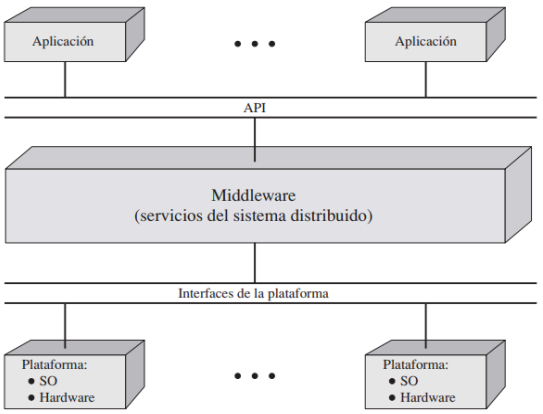


Figura 14.9. Visión lógica del middleware.

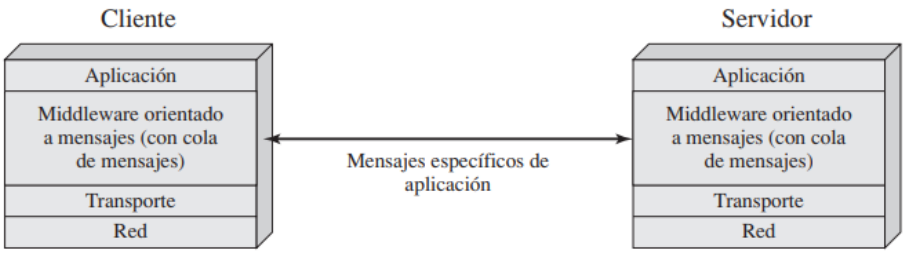
Existen distintos productos middleware, pero todos se basan en uno o dos mecanismos:

- Paso de mensajes
- Llamadas a procedimientos remotos

Paso de Mensajes Distribuidos

En los sistemas con procesamiento distribuido, las computadoras no comparten la memoria principal; cada una es un sistema aislado.
Para **comunicar procesos se utilizan técnicas que se basan en el paso de mensaje**, 2 de estas son las **técnicas más comunes**:

- Aplicación directa de los mensajes
- Llamadas a procedimientos remotos



(a) Middlewarw orientado a mensajes (con cola de mensajes)

Un proceso cliente necesita algún servicio y envía un mensaje con la petición de servicio a un proceso servidor. El proceso servidor realiza la petición y envía un mensaje con la respuesta

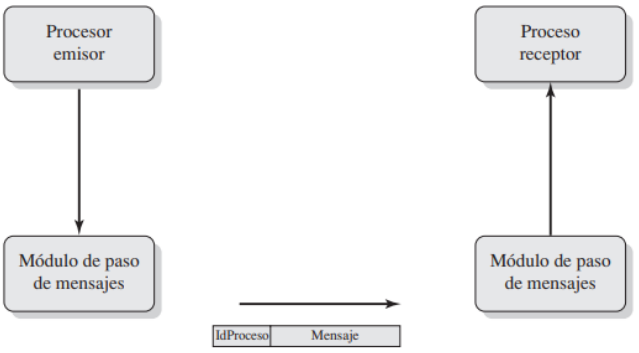


Figura 14.11. Primitivas básicas de paso de mensajes.

Figura 14.11 sugiere una implementación del paso de mensajes.
Los procesos hacen uso de los servicios de un módulo de paso de mensajes. Las peticiones de servicio se pueden expresar en términos de **primitivas** y **parámetros**, donde la **primitiva**, especifica la **función que se desea realizar** y los **parámetros** se utilizan para pasar los **datos e información de control**.
Ejemplos de primitivas:

- Send: Especifica un destinatario e incluye el contenido del mensaje
- Receive: Indica de quién se desea recibir un mensaje y proporciona un buffer donde se almacenará el mensaje entrante

Fiable vs No Fiable

Servicio fiable de paso de mensajes: Garantiza que la entrega es posible. Hacen uso de un protocolo de transporte fiable o de una lógica similar, y realizan comprobación de errores, acuse de recibo, retransmisión y reordenamiento de mensajes desordenados. Ya que se garantiza el envío, no es necesario informar al proceso emisor de que el mensaje ha sido enviado.
Servicio de paso de mensajes no fiable: Envía el mensaje por la red pero no se le indica ni el éxito ni el fracaso. Esta alternativa reduce enormemente la complejidad y la sobrecarga de procesamiento y comunicación del servicio de paso de mensajes.

Bloqueante vs No Bloqueante

Con las **primitivas no bloqueantes (asíncronas)**, no se suspende a un proceso como resultado de realizar un Send o Receive.

- Send no bloqueante → El control se devuelve inmediatamente al proceso emisor después de que el mensaje ha sido colocado en la cola para su transmisión o se ha realizado una copia. El proceso emisor puede continuar ejecutándose y hacer cambios en el mensaje, pero es responsable de asegurarse de que estos cambios no interfieran con la transmisión del mensaje.
- Receive no bloqueante → Permite que el proceso receptor continúe ejecutándose. El sistema operativo o el proceso receptor deben verificar periódicamente si ha llegado un mensaje. Cuando el mensaje está disponible, se informa al proceso receptor, generalmente mediante una interrupción o mediante una consulta periódica.

La **ventaja** de estas es que proporcionan un **uso eficiente y flexible de los servicios de paso de mensajes por parte de los procesos**. La **desventaja** es la **dificultad de chequear y depurar programas que utilicen estas primitivas**.

Las **primitivas bloqueantes**, si suspenden al proceso al realizar un Send o Receive.

- Send bloqueante → No devuelve el control al proceso emisor hasta que el mensaje ha sido transmitido (servicio no fiable) o hasta que el mensaje ha sido enviado y se ha recibido el acuse de recibo (servicio fiable).
- Receive bloqueante → No devuelve el control hasta que el mensaje ha sido situado en su correspondiente buffer.

Las primitivas bloqueantes, **resuelven las desventajas de las no bloqueantes**, por lo que esas son sus ventajas.

Llamadas a Procedimiento Remoto

Las llamadas a procedimiento remoto son una variante del modelo básico de paso de mensajes.
En esta variante, **se utilizan** las **llamadas a procedimiento para acceder a los servicios remotos**.
También, nos proporciona las siguientes **ventajas**:

1. Las llamadas a procedimiento son una abstracción ampliamente aceptada, utilizada y entendida.
2. El uso de llamadas a procedimiento remoto permite especificar las interfaces remotas como un conjunto de operaciones con nombre y tipos de datos dados. De esta forma, la interfaz se puede documentar claramente y los programas distribuidos pueden comprobar estáticamente errores en los tipos de datos.
3. Ya que se especifica una interfaz estandarizada y precisa, el código de comunicación para una aplicación se puede generar automáticamente.
4. Ya que se especifica una interfaz estandarizada y precisa, los desarrolladores pueden escribir módulos cliente y servidor que se pueden mover entre computadoras y sistemas operativos con pocas modificaciones y recodificaciones.

El mecanismo de llamadas a procedimiento remoto se puede ver como un refinamiento del paso de mensajes fiable y bloqueante.

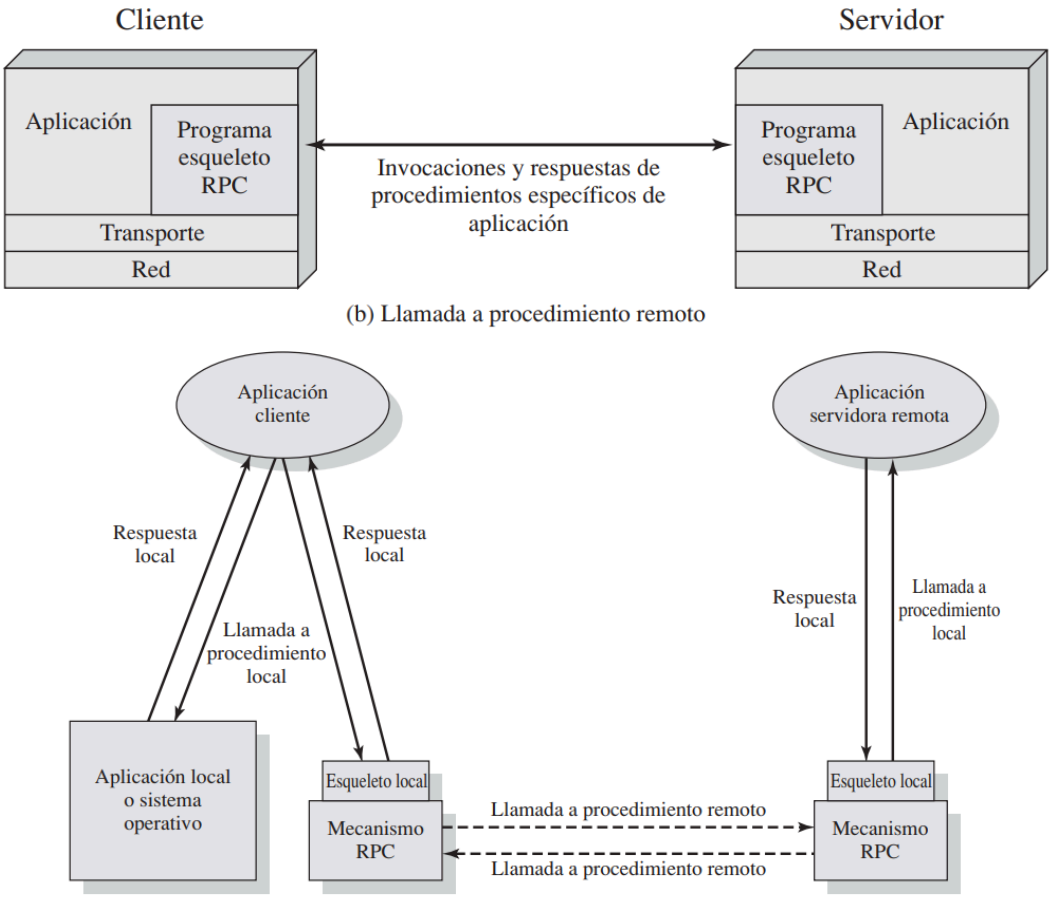


Figura 14.12. Mecanismo de llamadas a procedimiento remoto.

Clusters

Clusters: Grupo de **computadoras completas** e interconectadas, que trabajan juntas como un recurso de computación unificado y que pueden crear la ilusión de ser una única máquina. Cada computadora del Cluster se denomina **nodo**.
Son una alternativa al SMP y son **sistemas que proporcionan un alto rendimiento y una alta disponibilidad** y que son **atractivos para aplicaciones de servidor**.

Computadora completa: Sistema que puede ejecutar por sí mismo, aparte del cluster.

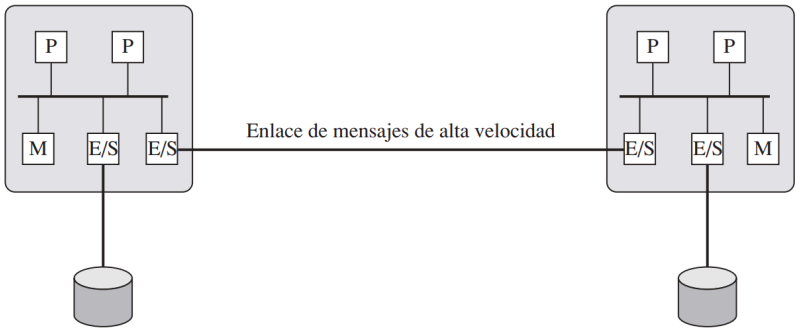
Estos clusters, buscan conseguir **4 objetivos de diseño**:

- **Escalabilidad absoluta**: Es posible crear un gran cluster que supere la potencia de incluso la mayor de las máquinas. Un cluster puede tener decenas o incluso centenas de máquinas, cada una de ellas un multiprocesador.
- **Escalabilidad incremental**: Un cluster se configura de tal manera que sea posible añadir nuevos sistemas al cluster en pequeños incrementos. Entonces, un usuario puede comenzar con un sistema pequeño y expandirlo según sus necesidades.
- **Alta disponibilidad**: Ya que cada nodo del cluster es una computadora en sí mismo, el fallo de uno de los nodos no significa pérdida del servicio.

- **Relación precio/prestaciones**: A través del uso de bloques de construcción es posible hacer un cluster con igual o mayor poder computacional que una única máquina mayor, con mucho menor coste.

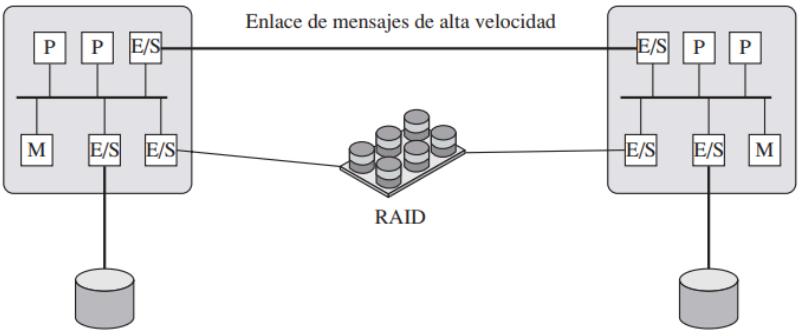
Configuración de los Clusters

Los Clusters se clasifican de varias maneras diferentes. La **configuración más sencilla** está basada en si las computadoras del cluster **comparten acceso a los discos**.



(a) Servidor en espera sin disco compartido

Cluster de dos nodos, con una interconexión a través de un enlace para el intercambio de mensajes y para coordinar la actividad del cluster.



(b) Disco compartido

Cluster con disco compartido. En este caso, sigue habiendo un enlace entre los nodos. Además, hay un subsistema de discos que está directamente enlazado con múltiples computadoras del cluster, el cual es un sistema RAID.

El uso de RAID o de algún sistema de redundancia similar, es común en los cluster para que la gran disponibilidad lograda por la presencia de múltiples computadoras, no se comprometa por un disco compartido, lo que podría ser un punto único de fallo.

Existen distintos **métodos de cluster**:

- **Pasivo en Espera**: Consiste en tener **una computadora realizando todo el proceso**, mientras que **otra permanece inactiva**, esperando tomar control en caso de fallo de la primaria.
  El sistema manda periódicamente un mensaje de «latido» a la máquina en espera. Si estos mensajes dejan de llegar, la máquina en espera asume que el servidor primario ha fallado y toma el control. Este enfoque **mejora la disponibilidad**, aunque **no se conoce normalmente como cluster**.
- **Servidor diferente**: Consiste en que cada **computadora tenga sus propios discos** y **no hay discos compartidos** entre los sistemas. Esta disposición **proporciona alto rendimiento** y **alta disponibilidad**.
  En este enfoque, es deseable tener la capacidad de recuperación de fallos (failover), lo que

significa que si una computadora falla durante la ejecución de una aplicación, otra computadora del cluster puede tomar el control y finalizar la aplicación.

- **Nada compartido** (**servidores conectados a discos comunes**): Los **discos comunes se particionan en volúmenes**, y cada volumen pertenece a una computadora.
  Si falla una computadora, el cluster se debe reconfigurar para que otra computadora tome posesión del volumen de la computadora que falló.
- **Disco compartido**: Consiste en tener **múltiples computadoras compartiendo los mismos discos al mismo tiempo**, de forma que cada computadora tiene acceso a todos los volúmenes de todos los discos.

Tabla 14.2. Métodos de *cluster*: beneficios y limitaciones.

Método de *Cluster*	Descripción	Beneficios	Limitaciones
Pasivo en Espera	En caso de fallo en el servidor primario, un servidor secundario toma control.	Fácil de implementar.	Alto coste debido a que el servidor secundario no está disponible para procesar otras tareas.
Secundario Activo	El servidor secundario también se utiliza para procesamiento de tareas	Coste reducido porque el servidor secundario puede ser utilizado para procesamiento.	Creciente complejidad.
Diferentes Servidores	Cada servidor tiene sus propios discos. Los datos se copian continuamente del servidor primario al secundario.	Alta disponibilidad.	Alta sobrecarga de red y de servidor debido a las operaciones de copia.
Servidores Conectados a Discos	Los servidores están unidos a los mismos discos, pero cada servidor posee sus propios discos. Si un servidor falla el otro servidor toma control de sus discos.	Sobrecarga de red y de servidores reducida debido a la eliminación de las operaciones de copia.	Normalmente requiere tecnologías de replicación de discos o RAID para compensar el riesgo de fallo de disco.
Servidores Comparten Discos	Varios servidores comparten acceso a disco de forma simultánea.	Baja sobrecarga de red y de servidores. Reducido riesgo de periodos de inactividad causados por fallos de disco.	Requiere software de gestión de cerrojos. Normalmente se utiliza con tecnologías de replicación de discos o RAID.

Arquitectura de un CLuster

Las **computadoras** están **conectadas a una LAN** de alta velocidad o conmutador hardware. **Cada computadora es capaz de operar independientemente**. Además, en **cada computadora está instalada una capa de software middleware que permite la operación del cluster**. El **middleware del cluster proporciona** una imagen única al usuario, conocida como **imagen única del sistema**.

Existen distintos **servicios y funciones deseables en un cluster**:

- **Un único punto de entrada**: Un usuario se autentifica en el cluster y no en una determinada computadora.
- **Una única jerarquía de ficheros**: Los usuarios ven una sola jerarquía de directorios bajo el mismo directorio raíz.
- **Un único punto de control**: Hay un nodo por defecto encargado de gestionar y controlar el cluster.

- **Una única red virtual**: Cualquier nodo puede acceder a cualquier otro punto del cluster, incluso si la configuración del cluster tienen múltiples redes interconectadas. Se opera sobre una única red virtual.
- **Un único espacio de memoria**: La memoria compartida distribuida permite a los programas compartir variables.
- **Un único sistema de control de trabajos**: Con un planificador de trabajos en el cluster, un usuario puede enviar su trabajo sin especificar la computadora que lo ejecutará.
- **Un único interfaz de usuario**: Todos los usuarios tienen un interfaz gráfico común, independientemente de la estación de trabajo que utilicen.
- **Un único espacio de E/S**: Cualquier nodo puede acceder remotamente a cualquier periférico de E/S o disco, sin conocer su localización física.
- **Un único espacio de procesos**: Se utiliza un esquema uniforme de identificación de procesos. Un proceso en cualquier nodo puede crear o se puede comunicar con cualquier otro proceso en un nodo remoto.
- **Puntos de control**: Esta función salva periódicamente el estado del proceso y los resultados de computación intermedios, para permitir recuperarse después de un fallo.
- **Migración de procesos**: Esta función permite el balanceado de cargas.

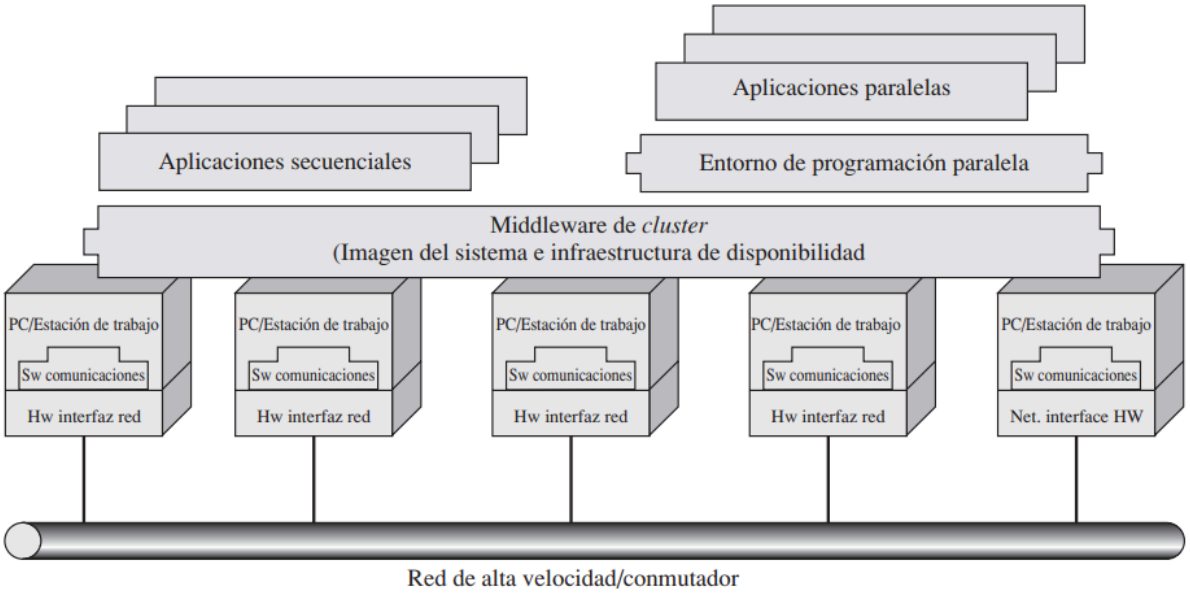


Figura 14.14. Arquitectura de computación *cluster* [BUYY99a].

Seguridad y Protección

Seguridad en los Sistemas

La seguridad de los sistemas puede ser de 2 maneras: Física y Lógica

Amenazas de Seguridad

La seguridad de los sistemas informáticos y de la red va dirigida a **cuatro requisitos básicos**:

- **Confidencialidad**: Requiere que la información de un sistema informático sólo se encuentre accesible para lectura para aquellas partes que estén autorizadas a este tipo de acceso.

- **Integridad**: Requiere que los contenidos de un sistema informático sólo podrán modificarse por las partes que se encuentran autorizadas.
- **Disponibilidad**: Requiere que los componentes de un sistema informático estén disponibles para todas aquellas partes autorizadas.
- **Autenticación**: Requiere que el sistema informático sea capaz de verificar la identidad de los usuarios.

TIPOS DE PELIGROS

Los tipos de ataques contra la seguridad del sistema o de la red se clasifican mejor considerando las funciones de un sistema informático como si se tratase de un proveedor de información. En general, existe un flujo de información desde una fuente, a un destino. Existen **4 categorías generales de ataques**:

- **Interrupción**: Se destruye un componente del sistema o se encuentra no disponible o utilizable. Es un ataque centrado en la **disponibilidad**. Ejemplo: Eliminación de una pieza de hardware o eliminación del sistema gestor de ficheros.
- **Intercepción**: Una parte no autorizada consiga acceso a un componente. Esto es un ataque dirigido hacia la **confidencialidad**. La parte no autorizada puede ser una persona, un programa o un ordenador
- **Modificación**: Un elemento no autorizado no sólo tiene acceso a un componente sino que también es capaz de modificarlo. Éste es un ataque que va dirigido hacia la **integridad**.
- **Fabricación**: Un elemento no autorizado inserta objetos extraños en el sistema. Estos son ataques contra la **autenticación**. Ejemplo: Inserción de mensajes externos en una red

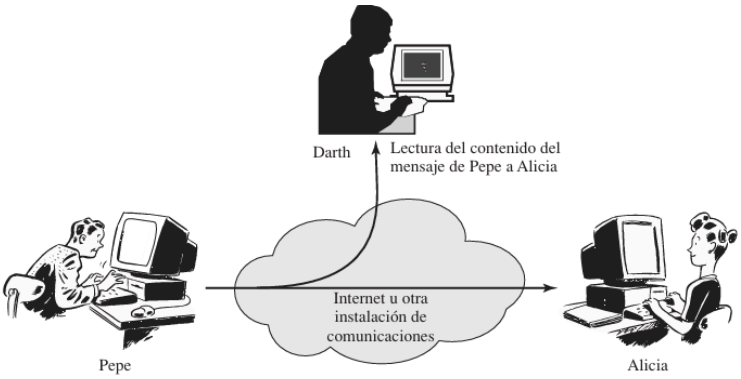
Componentes de un Sistema Informático

Los componentes de un sistema informático se pueden clasificar en:

- **Hardware**: El principal peligro del hardware de un sistema informático se encuentra en el área de la disponibilidad. El hardware es el componente más vulnerable a ataques y también es el menos accesible a una manipulación remota. Los principales peligros incluyen daño accidental o deliberado a los equipos, así como el robo. Para hacer frente a estos peligros se necesitan medidas de seguridad física y administrativa.
- **Software**: Existen diferentes tipos de peligros a tener en cuenta. Un peligro importante es el referente a la **disponibilidad**, ya que a menudo, este software es fácil de borrar, de la misma forma, también puede dañarse o alterarse hasta quedar inservible. Para mantener una alta disponibilidad, este software debe realizar **respaldos** (backups) y actualizarse a sus versiones más recientes.
  Una cuestión más complicada es la de modificación del software, el cual lo deja funcionar pero altera su comportamiento, implicando un peligro para la **integridad** y **autenticación** (Ejemplos: Virus informáticos).
  El último problema es el relativo a la **privacidad**, el cual aún no está resuelto.
- **Datos**: Los aspectos de seguridad relativos a los datos son muy amplios, incluyendo la disponibilidad, la privacidad y la integridad. El caso de la **disponibilidad**, se centra en la destrucción de ficheros de datos, que puede ocurrir de forma accidental o maliciosa.
  Respecto a la **privacidad**, encontramos la lectura no autorizada de ficheros de datos o bases de datos, además del análisis de datos que se manifiesta en bases de datos estadísticas.
  Para finalizar, la **integridad** de datos es un aspecto clave ya que la modificación de los ficheros

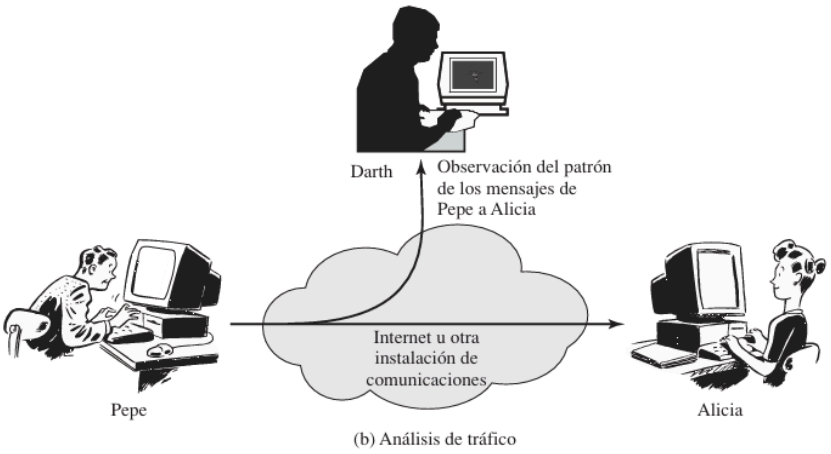
de datos puede llevar a una serie de consecuencias que van desde problemas menores hasta desastrosos.

- **Líneas de comunicaciones y redes**: Un mecanismo útil para la clasificación de los ataques son los términos de **ataques pasivos** y **ataques activos**.
  Ataques pasivos: Espionaje o monitorización de las transmisiones. El objetivo del oponente es **obtener información de qué se está transmitiendo**, pero no afectar a los recursos del mismo.
  Existen **2 tipos de ataque pasivos**:
  - **Lectura de los contenidos de los mensajes**: Lectura de un mensaje



(a) Lectura de los contenidos de los mensajes

  - **Ataques de análisis de tráfico**: Son más discretos. Supongamos que tenemos un **mecanismo para ocultar los contenidos de los mensajes** u otro tráfico de información de forma que los oponentes, intrusos si capturan los mensajes, no puedan extraer la información que contiene (Normalmente el cifrado), **un oponente** aún **podría observar cuáles son los patrones de estos mensajes**.



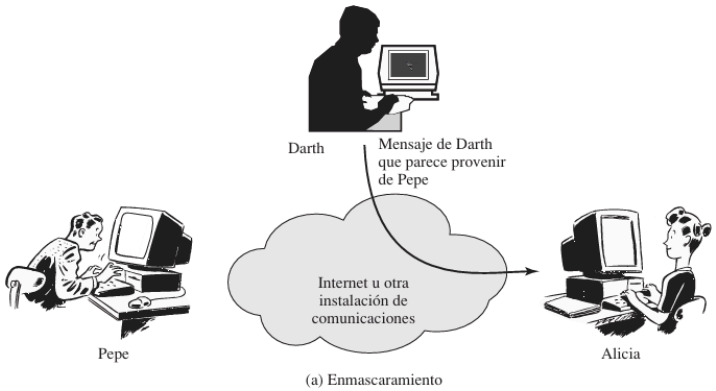
(b) Análisis de tráfico

Figura 16.3. Ataques pasivos.

Los ataques pasivos son muy **difíciles de detectar debido a que no implican ninguna alteración de los datos**.
El énfasis de seguridad que se hace sobre los ataques pasivos se centra en su prevención más que en su detección.

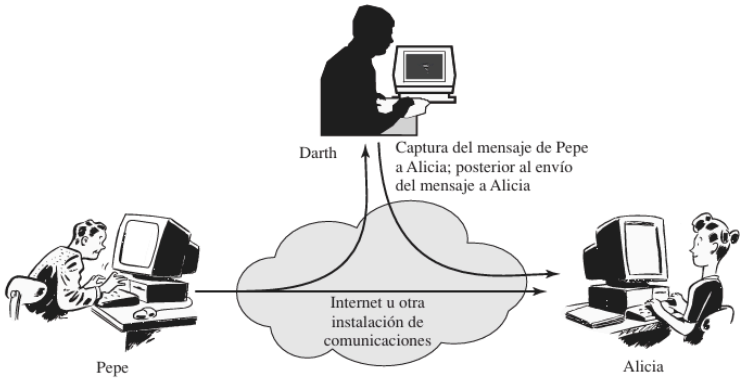
Ataques Activos: Implican algunas modificaciones en el flujo de datos o la creación de flujos de datos falsos. Se subdividen en **4 categorías**:

- **Enmascaramiento**: Ocurre cuando el elemento intenta hacerse pasar por otro diferente



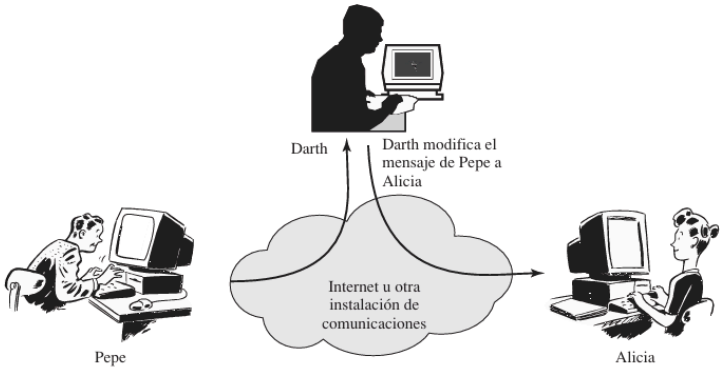
(a) Enmascaramiento

- **Reenvío**: Implica la captura pasiva de una unidad de datos y su posterior retransmisión para producir un efecto no autorizado.



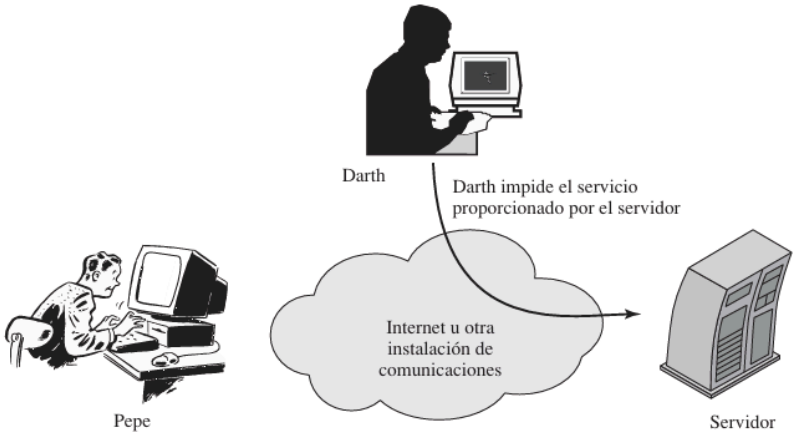
(b) Reenvío

- **Modificación de mensajes**: Significa que una parte de un mensaje válido se ha alterado, o que los mensajes se han borrado o reordenado, para producir un efecto no autorizado.



(c) Modificación de mensajes

- **Denegación de servicio**: Imposibilita el uso normal o la gestión de las instalaciones de comunicaciones. Se puede suprimir todos los mensajes dirigidos a un destino particular o desarticular toda la red sobrecargando con mensajes para degradar su rendimiento.



(d) Denegación de servicio

Los ataques activos presentan características opuestas a los ataques pasivos. Son bastante **más difíciles de prevenir de forma completa**, debido a que para poder hacerlo se requerirían protecciones físicas para todas las instalaciones de comunicaciones y todas las rutas.

Protección

La multiprogramación trajo consigo la posibilidad de compartir recursos entre los usuarios. Esta compartición implica no sólo procesador sino también:

- Memoria
- Dispositivos de E/S, como discos e impresoras
- Programas
- Datos

El S.O manejará diferentes niveles de protección sobre los recursos compartidos, entre ellos:

- **Sin protección alguna**. Apropiado por los procedimientos que son sensibles de ejecutar en instantes diferentes.
- **Aislamiento**. Implica que cada proceso opera de forma separada con otros procesos, sin compartición ni comunicación alguna.
- **Compartición** completa o sin compartición. El propietario del objeto declara si va a ser público o privado.
- **Compartición vía limitaciones de acceso**. El S.O verifica la permisibilidad de cada acceso por parte de cada usuario específico sobre cada objeto.
- **Acceso vía capacidades dinámicas**. Permite la creación dinámica de derechos de acceso a los objetos.
- **Uso limitado de un objeto**. Limita no sólo el acceso a un objeto sino también el uso que se puede realizar de dicho objeto.

Las medidas de control de acceso para un sistema de procesamiento de datos se encuadran en dos diferentes categorías: aquellas **asociadas al usuario** y aquellas **asociadas a los datos**.

CONTROL DE ACCESO ORIENTADO A USUARIO

La **técnica más común** para el control de acceso por usuario en un sistema compartido o en un servidor es el registro o conexión de usuario (**user log on**), se requiere el identificaron usuario (ID) y una contraseña (password).
Este **sistema ID/password** es un método poco fiable para proporcionar control de acceso a los usuarios, ya que estos pueden olvidar sus contraseñas y accidentalmente revelarlas. Los hackers son especialmente hábiles en adivinar los identificadores y contraseñas de usuarios especiales (Ej: Control del sistema, personal de gestión de sistemas).

El control de acceso de usuarios en un entorno distribuido se puede llevar a cabo de forma centralizada o descentralizada:

- **Centralizada**: La red proporciona un servicio de conexión, que determina a quién está permitido el uso de la red y con quién le está permitido conectarse.
- **Descentralizado**: Trata la red como un enlace de comunicación transparente, y el mecanismo de acceso habitual se realiza por parte del ordenador destino.

En muchas redes, en donde la red conecta diferentes ordenadores y únicamente proporciona mecanismos apropiados para el acceso entre un terminal y el ordenador, se puede utilizar el **control de acceso en dos niveles**:

1. Los ordenadores de forma particular pueden proporcionar un servicio de conexión para proteger los recursos y aplicaciones específicas de ese ordenador
2. La red en su conjunto puede proporcionar una protección de acceso restringido únicamente a los usuarios autorizados.

Control de Acceso Orientado a los Datos

Una vez que se ha tenido éxito en la conexión al sistema, el usuario ha obtenido acceso a uno o más conjuntos de ordenadores y sus respectivas aplicaciones.
Asociado con **cada usuario**, puede existir **un perfil que especifica las operaciones permitidas en los accesos a ficheros**.

Un modelo general para el control de acceso aplicado por un sistema de gestión de base datos o un sistema de ficheros es el denominado **matriz de acceso**. Este modelo contiene los siguientes **elementos básicos**:

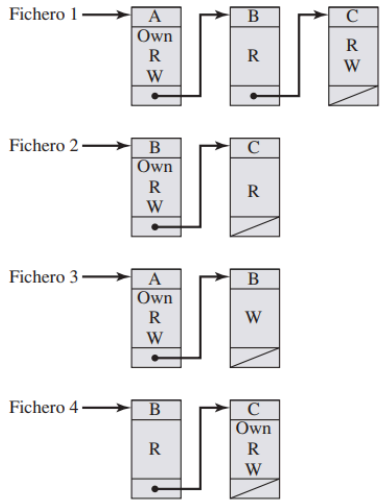
- **Sujeto**: Un elemento capaz de acceder a los objetos.
- **Objeto**: Todo elemento sobre el que se accede de forma controlada.
- **Derecho de acceso**: La forma en la cual un objeto es accedido por un sujeto (Ej: Lectura "R", Escritura "W").

	Fichero 1	Fichero 2	Fichero 3	Fichero 4	Account 1	Account 2
Usuario A	Own R W		Own R W		Información crédito	
Usuario B	R	Own R W	W	R	Información débito	Información crédito
Usuario C	R W	R		Own R W		Información débito

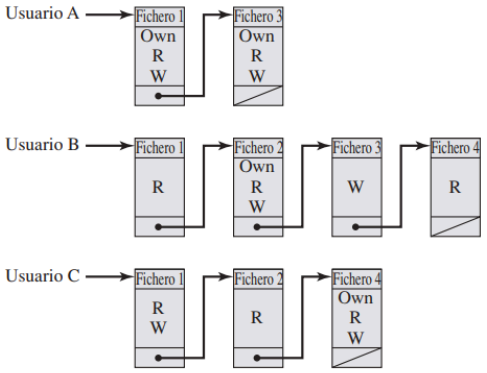
(a) Matriz de acceso

La matriz se puede descomponer por columnas, definiendo **listas de control de acceso**. De esta forma **por cada objeto**, hay **una lista de control de acceso** que muestra los usuarios y sus derechos de acceso. La descomposición por filas lleva a la definición de los **tickets de capacidades**. El usuario tiene un número de tickets y puede estar autorizado a cederlos a otros.
Debido a que los tickets pueden encontrarse dispersos a lo largo del sistema, representa un problema de seguridad mayor que las listas de control acceso. Particularmente, **un ticket no debe ser falsificable**.
Para mantener esto, el S.O se encarga de mantener los tickets de todos los usuarios almacenados en una región de memoria no accesible para los mismos.

Tickets de capacidades: Específica objetos y operaciones autorizadas para un determinado usuario.



(b) Listas de control de acceso para los ficheros de la parte (a)



(c) Listas de capacidades para los ficheros de la parte (a)

Intrusos

Uno de los dos peligros de seguridad más conocidos son los intrusos, generalmente denominados hackers o crackers. Se identifican tres clases de intrusos:

- **Enmascarado**: Un individuo que no está autorizado a utilizar un ordenador y que penetra en los controles de acceso del sistema para aprovecharse de una cuenta de usuario legítimo. Se trata habitualmente de un **usuario externo**.
- **Trasgresor**: Un usuario legítimo que accede a datos, programas, o recursos para los cuales dicho acceso no está autorizado, o estando autorizado para dicho acceso utilizar sus privilegios de forma maliciosa. Se trata habitualmente de un **usuario interno**.
- **Usuario clandestino**: Un usuario que sobrepasa el control de supervisión del sistema y usa dicho control para evadir la auditoría y el control de acceso o para suprimir la recogida de registros de acceso. Se puede tratar de un usuario externo o interno.

Técnicas de Intrusión

Ingeniería social
Prevención: Es el reto de la seguridad informática y una dura batalla desde siempre.
Detección: Se centra en el reconocimiento de los ataques, antes o justo después de que tengan éxito.

El objetivo de un intruso es ganar acceso a un sistema o incrementar el rango de sus privilegios de acceso a dicho sistema. Con el conocimiento de la contraseña de otro usuario, un intruso puede acceder al sistema y utilizar todos los privilegios acordes al usuario legítimo.
Habitualmente, un sistema debe mantener un **fichero que asocia las contraseñas con cada usuario autorizado**. Existen distintas formas de proteger el fichero de contraseñas:

- **Cifrado unidireccional**: El sistema almacena únicamente una forma cifrada de la contraseña de usuario.
- **Control acceso**: El acceso al fichero que contiene las contraseñas se encuentra limitado a una o muy pocas cuentas.

Para los crackers, existen distintas técnicas para conocer las contraseñas, entre ellas se destacan:

- **Mecanismos para intentar adivinar una contraseña**. Por ejemplo: Probar con el número de teléfono, documento, domicilio, etc.
- **Utilización de troyanos para sobrepasar las restricciones de acceso**. Es difícil de contrarrestar.
- **Pinchar la línea entre un usuario remoto y el sistema destino**. Es una cuestión de seguridad física, puede evitarse por medio de técnicas de cifrado en los enlaces.

Protección de Contraseñas

La primera línea de defensa contra los intrusos es el sistema de contraseñas. La contraseña sirve para autentificar el identificador de aquel que se está conectando al sistema. El **identificador** proporciona seguridad de la siguiente manera:

- El identificador determina si el usuario está autorizado a conseguir el acceso al sistema.
- El identificador determina los privilegios asociados al usuario.
- El identificador también se utiliza para un control de acceso discrecional.

Vulnerabilidad de las contraseñas

No resulta factible para un atacante utilizar una técnica simple de fuerza bruta para probar todas las posibles combinaciones de caracteres para descubrir una contraseña. En lugar de eso, los password crackers se basan en el hecho de que la mayoría de las personas eligen **contraseñas que son fácilmente adivinables**.
El programa simplemente tiene que probar el fichero de contraseñas contra aquellas palabras que parezcan más probables. Debido a que mucha gente utiliza contraseñas fácilmente adivinables (Incluso de menos de 3 caracteres).

Control de acceso

Una forma de protegerse de los ataques de contraseñas es denegar el acceso al oponente al fichero de contraseñas. Si la parte del fichero de contraseñas cifradas se encuentra accesible sólo para los usuarios con el nivel de privilegios adecuados, el oponente no podrá leerlo sin conocer previamente la contraseña del usuario privilegiado. De igual manera, existen fallos en este método, por lo que una estrategia más efectiva sería obligar a los usuarios a asignar contraseñas que sean difíciles de adivinar.

Estrategias de selección de contraseñas

Dejada a su **propia decisión**, muchos **usuarios eligen una contraseña** que es demasiado corta o demasiado **fácil de adivinar**. En el otro extremo, **si a los usuarios se les asignan contraseñas** consistentes en ocho caracteres de forma aleatoria, la adivinación de las contraseñas es imposible, pero resulta también imposible para la mayoría de los usuarios **recordar sus propias contraseñas**.
Para ello se utilizan **4 técnicas básicas**:

- **Educación de los usuarios**: No tiene muchas posibilidades de éxito en la mayoría de instalaciones, sobre todo donde hay demasiados usuarios, ya que muchos simplemente lo ignorarian.
- **Contraseñas generadas por el ordenador**: También tienen sus problemas. Si la naturaleza de la contraseña es demasiado aleatoria, los usuarios no la recordarán.
- **Verificación reactiva de las contraseñas**: Es una estrategia mediante la cual el sistema de forma periódica ejecuta su propio programa de adivinación para encontrar posibles contraseñas adivinables, cancelando cualquier contraseña que resulta adivinada y notificando

al usuario. Tiene varias **desventajas**, ya que requiere un **uso intensivo de los recursos** para realizar el trabajo correctamente, y además, todas las **contraseñas** que sean **frágiles** permanecen vulnerables **hasta que sean detectadas por el verificador**.

- **Verificación proactiva de las contraseñas**: Se les permite a los usuarios seleccionar su propia contraseña. Sin embargo, en el momento de la selección, el sistema prueba a ver si la contraseña está permitida, y si no es así, la rechaza (Por ejemplo: Sistema de reglas, es decir, la contraseña debe tener al menos ocho caracteres y una mayúscula). Es la alternativa más prometedora para mejorar el sistema de seguridad de contraseñas.

Detección de Intrusos

Inevitablemente, el mejor sistema para la prevención de intrusos fallará. Una segunda línea de defensa para el sistema es la **detección de intrusos**. La detección de intrusos **se basa en la suposición de que el comportamiento de un intruso difiere del de un usuario legítimo** de forma que puede ser cuantificado.

No se puede esperar que haya una distinción clara y exacta entre un ataque de un intruso y la utilización habitual de los recursos por parte del usuario autorizado, pero si podemos esperar que exista determinado **solapamiento**.

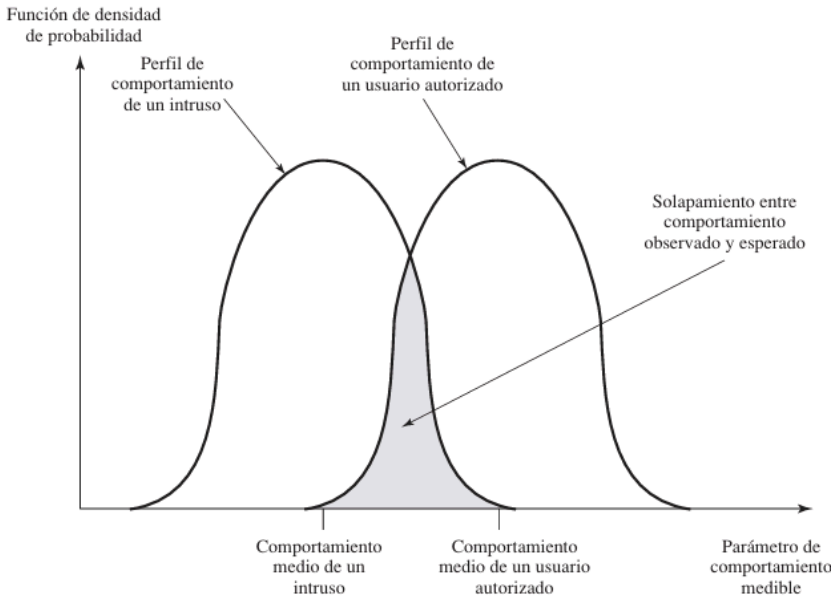


Figura 16.7 Perfiles de comportamiento de intrusos y usuarios autorizados.

Existen distintas técnicas para detección de intrusos:

1. **Detección estadística de anomalías**: Implica la **recolección de datos relativos al comportamiento de los usuarios** legítimos durante un período de tiempo. **Posteriormente se aplican una serie de tests** estadísticos a un nuevo **comportamiento** para determinar este es o no es de un usuario legítimo.
    a. **Detección por umbral**: Esta estrategia implica definición de umbrales, independientes del usuario, para la frecuencia de determinados eventos.
    b. **Basado en el perfil**: Se desarrolla un perfil de actividad por cada uno de los usuarios que, se utiliza para detectar cambios en el comportamiento de cada una de las cuentas de forma individual.
2. **Detección basada en reglas**: Implica un intento de definir un conjunto de reglas que se puedan utilizar para decidir si un comportamiento dado es o no el de un intruso.
    a. **Detección de anomalías**: Reglas desarrolladas para detectar la desviación de los patrones de usos previos.

b. **Identificación de penetración**: Un sistema experto que busca comportamiento sospechoso.

Una herramienta fundamental para detección de intrusos es el registro de auditoría. Se utilizan varios registros de las actividades, siendo dos de ellos los más utilizados:

- **Registros de auditoría nativos**: Prácticamente todos los sistemas operativos multiusuario incluyen un software de auditoría con el fin de registrar la actividad de los usuarios. La **ventaja** es que no se requiere ningún software adicional para recoger estos datos. La **desventaja** es que pueden no tener la información necesaria que puede requerirse
- **Registros de auditoría específicos para detección**: Se puede implementar una funcionalidad de recolección de datos que genere registros de auditoría que contienen información pensada únicamente para el sistema de detección de intrusos. La **ventaja** es que puede realizarse de forma independiente del vendedor e implantarse en una amplia variedad de sistemas. La **desventaja** es que implica tener una sobrecarga extra.

Cada uno de los registros de auditoría tiene siguientes los campos:

- **Sujeto**: Iniciador de la acción (Ejemplo: Terminal de usuario).
- **Acción**: Operación realizada por el sujeto utilizando un objeto (Ejemplo: Lectura, escritura).
- **Objeto**: El receptor de las acciones (Ejemplo: Ficheros, programas).
- Condiciones de excepción. Denota qué condiciones de excepción, si se dan, se lanzarían como respuesta.
- **Utilización de recursos**: Una lista de los elementos cuantitativos en los cuales los elementos calculan la cantidad de recursos utilizados (Ejemplo: Número de líneas impresas, número de registros leídos).
- **Sello de tiempo**: Un sello único de fecha y hora que identifica cuándo tuvo lugar esta acción.

Software Malicioso

Los tipos más sofisticados de amenazas para un sistema informático se encuentran presentes en los programas que explotan las vulnerabilidades de dicho sistema, estas amenazas son conocidas como **software malicioso** o **malware**.
El **malware** es software diseñado para causar daño o utilizar recursos de un ordenador, se encuentra escondido dentro de un programa enmascarado como software legítimo, y en algunos casos, se distribuye a sí mismo por medio del correo electrónico.

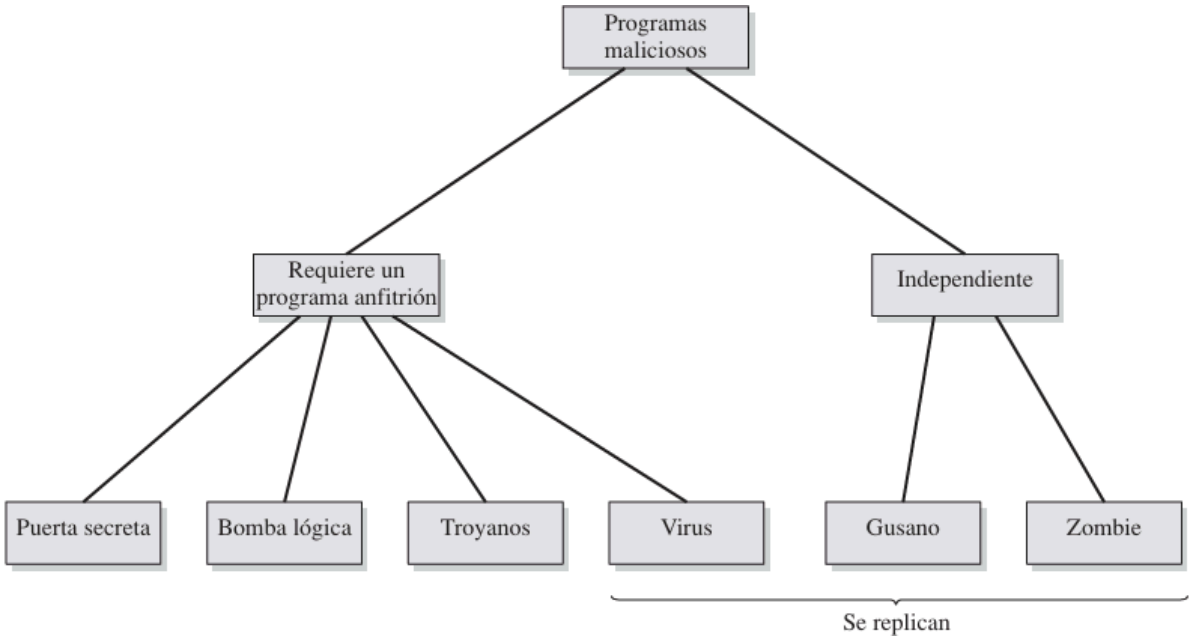
Programas Maliciosos

Estas amenazas se pueden dividir en dos diferentes categorías:

- **Aquellas que necesitan un programa anfitrión**:
  - Son esencialmente fragmentos de programa que no pueden existir de forma independiente sin una aplicación, utilidad o programa de sistema en particular.
- **Aquellas que son independientes**:
  - Son programas autónomos que pueden planificarse y ejecutarse por parte del sistema operativo.

También podemos realizar la diferencia entre los casos que no se pueden replicar y aquellos que sí. Aquellos que **no se pueden replicar** son **programas que deben activarse cuando el programa anfitrión se ejecuta para realizar una función específica**. Los casos que sí **se pueden replicar**, consisten en un fragmento de programa o programa independiente que cuando se ejecutan, **pueden**

producir una o más copias de sí mismo que se activarán posteriormente en el mismo o en otros sistemas.



Puerta Secreta: Una puerta secreta es un punto de entrada secreto dentro de un programa que permite a alguien que conoce la existencia de dicha puerta secreta tener el acceso sin utilizar los procedimientos de acceso de seguridad estándar.
Es muy difícil implementar controles por parte del sistema operativo para las puertas secretas.

Bomba Lógica: Una bomba lógica es un **código** insertado **dentro de un programa** legítimo que **explotará bajo ciertas condiciones**. Una vez activada, la bomba **puede alterar o borrar datos** o ficheros completos, causando que la máquina se detenga, o que se produzca algún daño.

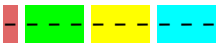
Troyano: Es un programa útil, o aparentemente útil, o mandato que contiene un código oculto que, al invocarse, realiza una función no deseada o dañina. Se utilizan para realizar tareas de forma indirecta que el usuario no autorizado no podría realizar directamente.
El programa puede parecer que realiza una tarea útil (por ejemplo un programa calculadora), pero puede también ir borrando silenciosamente los ficheros del usuario.

Virus: Es un programa que puede infectar otros programas modificándolos; las modificaciones incluyen la copia del programa virus, que puede a continuación infectar otros programas.
En un entorno de red, la posibilidad de acceder a aplicaciones y servicios de sistema de otro ordenador proporciona una infraestructura perfecta para la dispersión de los virus.

Gusano: Los gusanos **utilizan las conexiones de red para expandirse de un sistema a otro**. Una vez que se encuentran activos dentro de un sistema, un gusano de red se puede comportar como un virus informático, **puede implantar troyanos o realizar cualquier otro tipo de acciones destructivas**.
Para replicarse a sí mismo, un gusano de red utiliza algún tipo de vehículo de comunicación (Ejemplo: Herramientas de correo electrónico, capacidad de conexión remota).
Un gusano de red muestra las mismas características que un virus informático: una fase latente, una fase de preparación, una fase de activación y una fase de ejecución.
En un sistema **multiprogramado**, **puede disfrazar su presencia** renombrándose **como un proceso** de sistema o utilizando cualquier otro nombre que no resulte sospechoso para el operador de sistema.

Zombie: Un programa zombie toma el control de otro ordenador conectado a Internet y posteriormente utiliza el mismo para lanzar ataques que son difíciles de trazar como provenientes del creador del zombie. **Se utilizan** habitualmente para **ataques de denegación de servicio**, habitualmente contra sitios web que son sus objetivos.
Un zombie se implanta en centenares de ordenadores pertenecientes a terceras partes, que desconocen su existencia, y posteriormente utiliza todos estos puntos de acceso para derribar el sitio web en cuestión lanzando un tráfico de red intenso.

Permisos en Linux



Los permisos son: W-R-E (Escribir, Leer, Ejecutar)
Rojo: Tipo de archivo
Verde: Propietario archivo
Amarillo: Grupos
Celeste: Otros

Lista de Accesos: Los permisos están en los archivos, por ejemplo Linux y Windows la utilizan.
Lista de Capacidades: Más común en sistemas con gran cantidad de usuarios