MÓDULO: 6

Sistema de Gestión de Entrada / Salida

CONTENIDO:

- Conceptos sobre el Sistema de Gestión de Entrada / Salida.
- Administración del Hardware de Entrada / Salida.
- El Software de Entrada / Salida.

OBJETIVO DEL MÓDULO: Describir las distintas interfases que debe soportar el S.O. para poder administrar los dispositivos periféricos.

OBJETIVOS DEL APRENDIZAJE: Después de estudiar éste módulo, el alumno deberá estar en condiciones de:

- Explicar y reconocer las distintas estructuras o módulos de E/S.
- Conocer las funciones del Sistema de Gestión de Entrada / Salida.
- Comprender y explicar la organización de las interfases de E/S.
- Conocer y explicar la terminología específica empleada en éste módulo.
- Entender los mecanismos de vinculación utilizados en los accesos a los dispositivos.

Metas:

Empezamos este tema con una breve discusión de la estructura y función de un módulo de E/S, seguidos de una visión de los dispositivos externos. Luego, se observan las distintas formas en las cuales puede ser ejecutada una función de E/S en forma cooperativa con el procesador y la memoria central: la interfase de E/S interna. Finalmente es examinado el Software, la interfase de E/S externa, entre el módulo de E/S, el mundo externo y los servicios del S.O. y por último ejemplificamos un Sistema de E/S utilizado en un mainframe.

6. Administración de la Entrada/Salida (I/O Scheduler)

Convenimos que el término ***entrada*** se refiere a que el procesador[1] o Memoria Central recibe datos del exterior y ***salida*** que envían o transfieren datos a los dispositivos periféricos.

6.0 Introducción

Además del procesador y el conjunto de los módulos de memoria, el tercer elemento clave de un sistema de computación es el conjunto de los módulos de Entrada/Salida. Cada módulo se comunica con el bus o computador central y controla uno o más dispositivos periféricos. Un módulo de E/S no es un conector mecánico que cablea el dispositivo al bus del sistema. Al contrario, el módulo de E/S contiene "inteligencia" provista por un procesador que emplea una lógica para ejecutar un función comunicacional entre el dispositivo periférico y el bus del sistema.

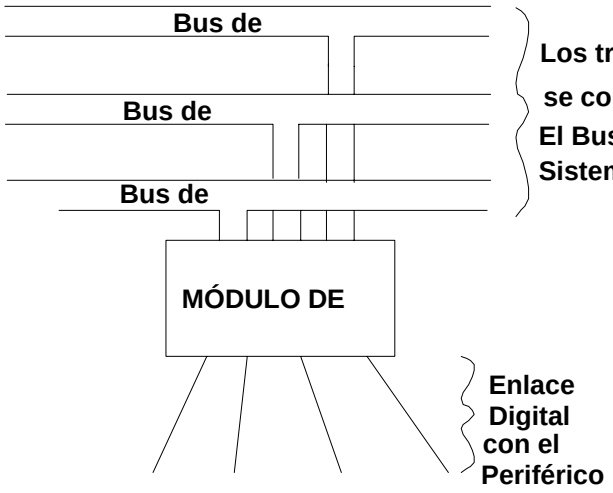


Figura 6.01a - Modelo genérico del Módulo de E/S

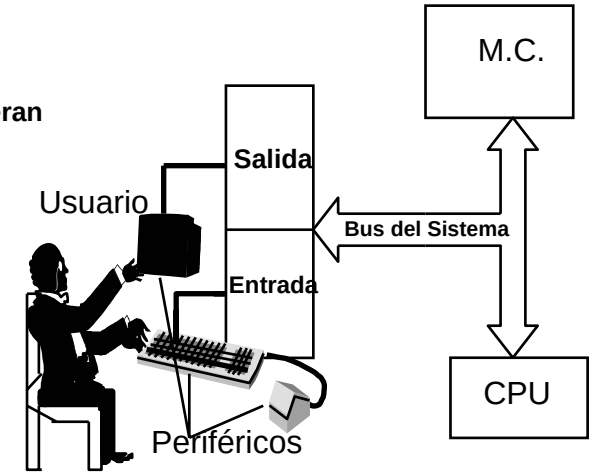


Figura 6.01b - Modelo simplificado del Módulo de E/S

Las razones del porqué los periféricos no se conectan en forma directa al bus del sistema son las siguientes:

* Hay una amplia variedad de periféricos con varios métodos de operación. Sería impráctico incorporar la lógica necesaria dentro del procesador para controlar un rango de dispositivos uno por uno. Además si apareciera un nuevo tipo de dispositivo, este no se podría conectar.
* La velocidad de transferencia de datos de los periféricos es a menudo bastante menor que la de la memoria central o del procesador. Luego no resulta práctico utilizar un bus de alta velocidad para comunicarse directamente con el dispositivo periférico.
* Los periféricos generalmente emplean distintos formatos de datos y longitudes de palabra que los utilizados en el computador al que se conecta.

En consecuencia, es necesario un módulo de E/S. Este tiene dos funciones principales (Figura 6.01) desde el punto de vista del Hardware:

* Proveer una interfase con el procesador y la memoria central vía el bus del sistema.
* Proveer una interfase a uno o más dispositivos periféricos con vínculos especialmente diseñados.

El Sistema de Gestión de Entrada / Salida

El Sistema de Gestión de la Entrada / Salida es la parte del S.O. que se ocupa de la organización, administración y operación de los dispositivos de E/S. Entendemos por dispositivos de entrada - salida a los periféricos con su respectiva unidad de control, conectados a los controladores y a los **puertos** (Ports) o canales que ofician de interfases entre el bus del sistema y los periféricos.

[1] Recordemos que usaremos la palabra procesador y CPU como sinónimos al igual que E/S o I/O para la Entrada/Salida.

Cada periférico tiene requerimientos específicos de programación, de temporización y de funcionamiento electromecánico.

La gestión de E/S debe ser eficiente, segura, confiable, abstracta, uniforme, etc. y actúa como interfase entre el programa en ejecución y los dispositivos, proveyendo una abstracción de las complejas operaciones y controles entre ellos.

Las actividades que realiza el sistema de E/S son concurrentes con el procesador y son autónomas pero no autosuficientes ya que requieren de la colaboración de otros módulos del S.O. (File System y administrador de memoria central).

Las operaciones de E/S funcionan concurrentemente, dado que se pueden iniciar una serie de operaciones independientemente de que se completen otras en progreso ya iniciadas. Estas tareas concurrentes requieren de un adecuado planeamiento para que no ocurran deadlocks.

6.1. Funciones del administrador de Entrada / Salida (E/S)

- Mantener un registro de estado de los dispositivos periféricos, controladores y canales y de los programas que controlan a estos dispositivos.
- Determinar una técnica de asignación mediante un procedimiento de vinculación (Binding).
- Controlar todos los Dispositivos de E/S (emitir comandos, capturar interrupciones y manejar errores generados por los dispositivos, inicializar los dispositivos en el arranque y en cada operación, procesar, cancelar y sincronizar las operaciones, etc.)
- Proveer una interfase entre los dispositivos y el resto de los Sistemas (Lógicamente abstracta, independiente, simple y fácil de usar, la misma para todos los dispositivos, etc.)
- Organizar, administrar y operar la transferencia de los datos y la información desde y a los dispositivos, ya sea temporal o definitivamente en forma eficiente.
  Conjuntamente con el Sistema de Gestión de Archivos, manejar los datos almacenados en los soportes, que implica la conversión de formatos, factores de bloqueos, ediciones parciales, apertura y cierre de archivos, almacenamiento y recuperación de la información. Manejar las colas de impresión (Spooler), etc..
- Proveer un mecanismo de protecciones y control de accesos.

6.2. Módulos de E/S y la estructura del módulo de E/S

6.2.1 Funciones del Módulo

Un módulo de E/S es la entidad dentro del computador que tiene la responsabilidad de controlar uno o más dispositivos externos e intercambiar los datos entre esos dispositivos y la memoria central y/o los registros de la CPU. En consecuencia, el módulo de E/S debe tener una interfase interna al computador (al procesador y la memoria central) de alta velocidad y una interfase externa al computador (al dispositivo externo).

Las funciones primordiales o requerimientos para un módulo de E/S caen dentro de las siguientes categorías:

* Control y Temporización
* Comunicación con el procesador
* Comunicación con el dispositivo
* Amortiguación (buffering) de datos
* Detección de errores.
* Manejo de Interrupciones en el bajo nivel.

Durante cualquier período o instante, el procesador puede comunicarse con uno o más dispositivos externos (en rigor, con uno o más módulos de E/S) de manera impredecible, dependiendo de las necesidades de E/S de los programas que están ejecutando. Los recursos internos, como la memoria o el bus del sistema, deben ser compartidos entre un número de actividades incluyendo a la E/S de datos. Luego, la función de E/S debe incluir un requerimiento de ***control y temporización***, para coordinar el flujo del tráfico de datos y señales entre los recursos internos y los dispositivos externos. Por ejemplo, el controlar la transferencia de datos desde un dispositivo externo y el procesador, puede incluir los siguientes pasos:

1. El procesador interroga al módulo de E/S para verificar el estado (status) del dispositivo conectado.
2. El módulo de E/S devuelve el estado del dispositivo.

3. Si el dispositivo está operable y listo para transmitir, el procesador solicita la transferencia de datos, por medio de un comando al módulo de E/S.
4. El módulo de E/S obtiene una unidad de datos (por ejemplo, 8 ó 16 bits) del dispositivo externo.
5. Los datos son transferidos desde el módulo de E/S al procesador.

Si el sistema utiliza un bus, cada una de las interacciones entre el procesador y el módulo de E/S requieren de una o más arbitraciones del bus.

El simplificado escenario precedente también ilustra que el módulo de E/S debe tener la capacidad de ocuparse de la comunicación con el procesador y el dispositivo externo. La comunicación con el procesador incluye:

* ***Descodificación del comando***: El módulo de E/S acepta comandos desde el procesador. Estos comandos generalmente son enviados como señales a través del bus de control. Por ejemplo, un módulo de E/S para un controlador de disco podría aceptar los siguientes comandos: READ SECTOR, WRITE SECTOR, SEEK número de pista, y SCAN record ID (rastrear identificación de registro). Cada uno de los dos últimos comandos incluye un parámetro que es enviado en el bus de datos.
* ***Datos***: Los datos son intercambiados entre el procesador y el módulo de E/S a través del bus de datos.
* ***Informe de Estados***: Por ser los periféricos tan lentos, resulta importante saber el estado del módulo de E/S. Por ejemplo, si a un módulo de E/S se lo invita a enviar datos al procesador (read), puede no estar listo para hacerlo por estar todavía ocupado en el comando previo de E/S. Este hecho puede ser informado con una señal de status en el bus de control. Las señales de estados más comunes son BUSY y READY. Puede haber también señales para informar varias condiciones de error.
* ***Reconocimiento de direcciones***: De la misma forma que cada palabra en la memoria tiene una dirección, también la tiene cada dispositivo de E/S. En consecuencia, un módulo de E/S debe reconocer la dirección única asociada a cada periférico que controla.

Por otro lado, el módulo de E/S debe ser capaz de realizar una ***comunicación con el dispositivo***. Esta comunicación incluye comandos, información de estado, y datos (Figura 6.02 y 6.07).

Dispositivo	Comportamiento		Velocidad (Kbyte/s)
Teclado	Entrada	Humano	0,01
Ratón	Entrada	Humano	0,02
Input de Voz	Entrada	Humano	0,02
Scanner	Entrada	Humano	200
Output de Voz	Salida	Humano	0,6
Impresora de Línea	Salida	Humano	1
Impresora Láser	Salida	Humano	100
Pantalla Gráfica	Salida	Humano	30.000
CPU a un buffer de cuadro	Salida	Humano	200
Terminal en Red	Entrada o Salida	Máquina	0,05
LAN	Entrada o Salida	Máquina	200
Disco Optico	Almacenamiento	Máquina	500
Cinta Magnética	Almacenamiento	Máquina	2.000
Disco Magnético	Almacenamiento	Máquina	2.000

TABLA 6.1 Cuadro demostrativo de Velocidades e interfases de dispositivos

Una tarea esencial del un módulo de E/S es la amortiguación de velocidades relativas ***(buffering)*** de datos. La necesidad de esta función emerge de la Tabla 6.1. Mientras que la velocidad de transferencia hacia y desde la memoria o el procesador es bastante alta. La velocidad es varios órdenes de magnitud inferior para la mayoría de los dispositivos periféricos. Los datos provenientes de la memoria central son enviados al módulo de E/S en ráfagas (burst) rápidas. Los datos son almacenados temporariamente (buffered) en el módulo de E/S y luego enviados al dispositivo periférico a su velocidad de datos. En la dirección opuesta, los datos son también almacenados temporariamente para no involucrar a la memoria en una operación de transferencia lenta. Luego, el módulo de E/S debe ser capaz de operar a ambas velocidades (de la memoria o procesador y del dispositivo).

Un módulo de E/S es a menudo también responsable de la ***detección de errores*** y subsecuentemente informar al procesador de dichos errores. Una clase de error puede ser el mal funcionamiento, mecánico o eléctrico, reportado por el dispositivo (por ejemplo, papel atrancado (paper jam), sector de disco "malo", etc.). Otra clase de error consiste en los cambios no intencionales de los patrones de bits tal como son transmitidos desde el dispositivo al módulo de E/S. Alguna forma de código de detección de errores es a menudo utilizado para detectar los errores de transmisión. El ejemplo más común es utilizar un bit de paridad anexado a cada carácter de datos. El octavo bit del código ASCII se

establece de manera tal que el total de "unos" en el byte es par (o paridad par) o impar (paridad impar). Cuando se recibe un byte, el módulo de E/S verifica la paridad para determinar si un error ha ocurrido (aún bajo este esquema, un doble error no sería detectado pero es poco frecuente).

Finalmente, el manejo de interrupciones al mas bajo nivel debe ser realizado por este módulo.

6.2.2 Estructura del Módulo de E/S

Los módulos de E/S varían considerablemente en complejidad y en el número de dispositivos externos que controlan. Haremos una somera descripción aquí. (Ver la sección sobre el chip Intel 8255A). Las Figuras 6.02 y 6.07 proveen un diagrama de bloques general para un módulo de E/S. El módulo se conecta al resto del computador a través de un conjunto de líneas de señal (líneas del bus del sistema) como se indica en las figuras 6.03, 6.05 y 6.06. Los datos que se transfieren desde o hacia el módulo se almacenan temporariamente en uno o más registros de datos (Buffers). También puede haber uno o más registros de Estados para proveer la información actual del estado. Un registro de status puede también funcionar como un registro de control, para aceptar información de control detallada desde el procesador. La lógica dentro del módulo interactúa con el procesador vía un conjunto de líneas del control. Estas son utilizadas por el procesador para emitir comandos al módulo de E/S. Algunas de las líneas de control pueden ser utilizadas por el módulo de E/S (por ejemplo, para las señales de arbitración de bus y señales de status). El módulo debe ser también capaz de reconocer y generar direcciones asociadas con los dispositivos que controla. Cada módulo de E/S tiene una dirección única o, si controla más de un dispositivo externo, un único conjunto de direcciones. Finalmente el módulo de E/S contiene una lógica específica para dialogar (to interface) con cada dispositivo que controla.

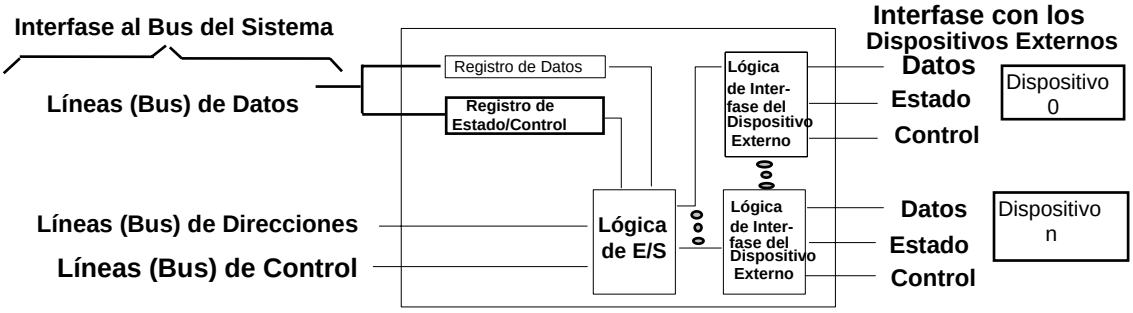


Figura 6.02 - Diagrama de Bloques de un Módulo de E/S

Un módulo de E/S funciona para permitir al procesador ver un amplio rango de dispositivos en una forma simple. Hay un espectro de capacidades que pueden ser provistas. El módulo de E/S esconde los detalles de temporizaciones, formatos, y la electromecánica de un dispositivo externo de manera tal que el procesador pueda operar en términos de simples comandos READ y WRITE, y posiblemente los comandos OPEN y CLOSE de archivos. En su forma más simple, el módulo de E/S puede aún dejar mucho del trabajo de controlar un dispositivo visible al procesador (como ser rebobinar una cinta).

Un módulo de E/S que carga con la mayoría de los detalles del procesamiento, presentando una interfase de alto nivel a la CPU, se conoce normalmente como un ***canal* o un *procesador de E/S.*** Un módulo de E/S que es bastante primitivo y requiere un control detallado, se conoce comúnmente como un ***controlador de E/S* o un *controlador de dispositivo***. Los controladores de dispositivos se ven normalmente en microcomputadores, mientras que los canales de E/S son utilizados en los mainframes, y en los minicomputadores se observa una mezcla.

Cuando un módulo de E/S es complejo (por ejemplo un canal de E/S), es usualmente particionado en la forma de un canal de E/S controlando uno o más controladores de E/S. Un ejemplo de esto se ve en la figura 6.09.

En lo que sigue, utilizaremos el término genérico de ***módulo de E/S*** para que no haya confusión y usaremos términos específicos cuando sea necesario.

6.3. Las operaciones del hardware de Entrada / Salida.

Se destacan dos características en las operaciones de los dispositivos de E/S:

1. Operación asincrónica
2. Diferencias de velocidades.

6.3.1. Operación Asincrónica:

Los dispositivos funcionan independientemente del procesador y no utilizan el reloj del procesador, sino el suyo propio o alguna acción del usuario (Por ej.: una lectura en un disco, apretar teclas, movimiento

del ratón, etc.). Esta temporización no guarda ninguna relación con el reloj del sistema (CPU). Por otro lado, la llegada de datos y los tiempos de transferencia de la E/S son generalmente impredecibles. Esto constituye una operación asincrónica manejado por las señales de control que son generados por los dispositivos y sus interfases. A éstas señales se las conoce como ***señales de dialogo (handshaking***) que son intercambiadas entre el controlador - dispositivo y el procesador antes de comenzar y al finalizar cada operación de E/S.

También el S.O. utilizará este protocolo de dialogo para consultar el estado de los dispositivos y verificar la vinculación de los mismos e inicializar la correspondiente operación solicitada por el proceso en ejecución. Si alguno de éstos no están disponibles, el proceso deberá esperar (estado del proceso: INACTIVO).

Mientras se efectúan todos los pasos para obtener y transferir los datos, o sea, completar la operación, el proceso estará en estado bloqueado (Wait for I/O). Durante esta situación el S.O. puede seleccionar otro proceso de la cola de listos para ejecutar y ponerle en uso del procesador si la política así lo permite (Con sustitución o preemptive) sino deberá permanecer en ***iddle*** (ocioso) esperando a que se complete la operación de E/S si la política es *sin sustitución*.

Cuando finaliza la operación de E/S en curso, las señales de diálogo y las interrupciones le comunican al S.O. informándole que se ha completado dicha operación y la misma debe ser atendida.

6.3.2. Diferencias de Velocidades.

Nos referimos a las diferencias de velocidades del procesador o Memoria Central con respecto a las operaciones de Entrada / Salida.

Las velocidades de los periféricos están en relación con las actividades electromecánicas que realizan y del mundo externo con el que se comunican (Usuarios, impresos, visualizadores, etc.).

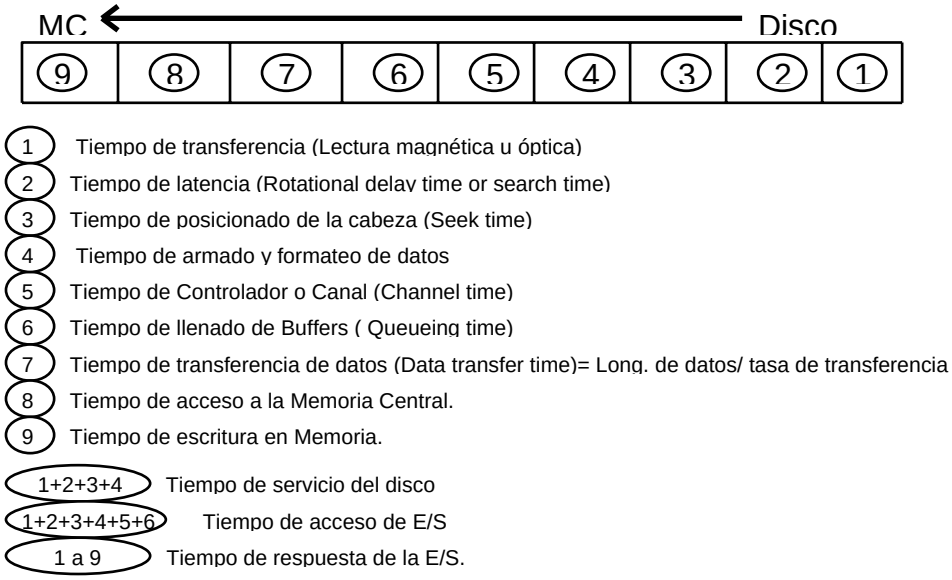


Figura 6.03 Distintos tiempos involucrados en una entrada (lectura de Disco)

La velocidad relativa del procesador (que operan en el rango de los nanosegundos) y los dispositivos (rango de milisegundos para los que presentan caracteres en pantallas o impresoras y de varios microsegundos en el caso de las comunicaciones por fibras ópticas) es del orden de 10^6 veces o mayor. Esto genera la necesidad de amortiguar las diferencias de velocidades a través de buffers cuando se realizan las transferencias de datos y a ésta técnica se la conoce como ***buffering*** *que se estudiara mas abajo* en este módulo en el punto 6.4.5.

Los dispositivos lentos, generalmente son controlados por el procesador, mientras que los de alta velocidad, lo hace un procesador especializado llamado ***canal de E/S.*** Es posible que la transferencia de datos en bloques lo realice un procesador especializado llamado ***DMA (Direct Memory Access)*** que estudiaremos más detalladamente en el punto 6.5.3.

Proponemos, como ejemplo en la Fig. 6.03, los distintos tiempos que intervienen en una lectura de disco y la transferencia de datos a la memoria central.

6.4. Los Dispositivos y sus Interfases (el Hardware de E/S)

6.4.1. Dispositivos de Entrada / Salida.

Según Tanenbaum, los dispositivos se pueden dividir en tres grandes grupos:

1. Dispositivos orientados a bloques: son los que manipulan bloques de datos de tamaño fijo (Ejemplo: 256 a 2048 Bytes) en forma independiente. Cada bloque tiene asociado una dirección (Por ej.: Discos: Flexibles, Duros, magneto-ópticos, ópticos).
2. Dispositivos orientados a Caracteres: son los que manipulan cadenas de caracteres o bytes sin tener en cuenta ninguna estructura predeterminada. No tienen asociados ninguna dirección ni operaciones de búsqueda (Por ej.: impresoras, visualizadores, teclados, consolas, modems, etc.)
3. Otros Dispositivos: no tienen direcciones ni bloques ni caracteres. (por ej.: relojes, graficadores bit-mapeados, ratones, pantallas en modo gráfico, microfilms, robots, sistema vocal, plotters, facsímiles, etc.)

Los dispositivos están compuestos por una parte electromecánica y otra electrónica (***unidad de control del dispositivo*** o adaptador o manejador - handler). Esta unidad de control, generalmente dispone de un microprocesador que oficia de la lógica programada para su operación. Cada dispositivo dispone de una interfase (Bus de datos y Bus de Control) para conectarse con el controlador del que depende su operación.

Las ventajas de utilizar microprocesadores en éstas unidades radica en: Disminución de tamaño, Menores consumos, Mayores velocidades, Gran adaptabilidad, Multifunción, Programabilidad, etc.

- Los dispositivos pueden estar conectados “on - line” (en línea controlado por el proceso), u “off - line” (fuera de línea).

En el caso del Sistema de Gestión de E/S sólo interesan los que están en línea.

- Los datos y la información cambian de soportes y de códigos.

Desde el punto de vista del S.O.:

- Interesa la organización y administración de las operaciones de entrada - salida en cuanto a su programación y no de su funcionamiento interno.
- También interesa los requerimientos efectuados por los Usuarios y sus programas y de los servicios de E/S que brinda el S.O. a esos requerimientos.

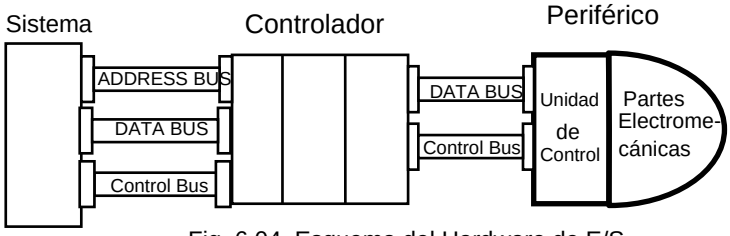


Fig. 6.04 Esquema del Hardware de E/S

6.4.2. Controlador, Adaptador o Interfase de Entrada - Salida.

Es la interfase del sistema de E/S con el S.O. a nivel de software. Se ocupa de convertir el flujo de bits en bloques de Bytes (en los dispositivos orientados a bloques, Por ej.: Discos) o caracteres (Byte) en los dispositivos orientados a caracteres. El sistema operativo se comunica con el controlador, no con el dispositivo. Por tanto, existen dos tipos de interfases: controlador-dispositivo y CPU-controlador.

Controla uno o más Unidades de periféricos del mismo tipo. Generalmente se interconecta mediante una Tarjeta de circuito impreso con el bus de sistema.

Las placas madres (motherboard) suelen tener diferentes zócalos (slots) para poder agregar controladores de diferentes tipos y funciones; por ejemplo zócalos AGP (Accelerated Graphics Port), PCI (Peripheral Component Interconnect Bus), ISA (Industry Standard Architecture), EISA, etc. Todos estos controladores responden a normas específicas.

Un controlador básicamente está compuesto por tres partes:

1. Una interfase con el bus del sistema.
2. Un controlador propiamente dicho (Lógica programada).
3. Una interfase con el dispositivo (Uno o más).

La primera adapta el bus del sistema a la del bus interno del controlador, ofreciendo una abstracción de una interfase homogénea para todos los controladores conectados. Además ésta interfase provee una independencia con respecto a los distintos tipos de dispositivos que se conectan al computador. Esta abstracción se logra mediante la integración de Hardware más Software de E/S. En particular se basan en

Normas o Standares como ser PCI (Peripherical Component Interconnect de Intel), o ISA (Industrial Standard Architecture), etc.

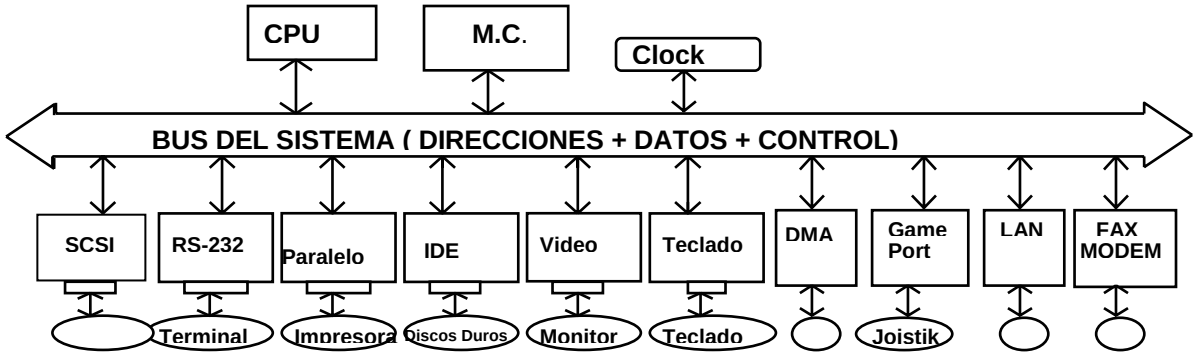


FIG. 6.05 Ejemplo de las distintas interfases disponibles para interconectar dispositivos de E/S.

La segunda etapa es la que oficia de interfase con el S.O. y provee la separación lógica - física y las abstracción de las operaciones de E/S y sus servicios.

La tercera, adapta el dispositivo al sistema, provocando que el mismo sea dependiente del controlador por los niveles de señales y las órdenes que lo hacen funcionar.

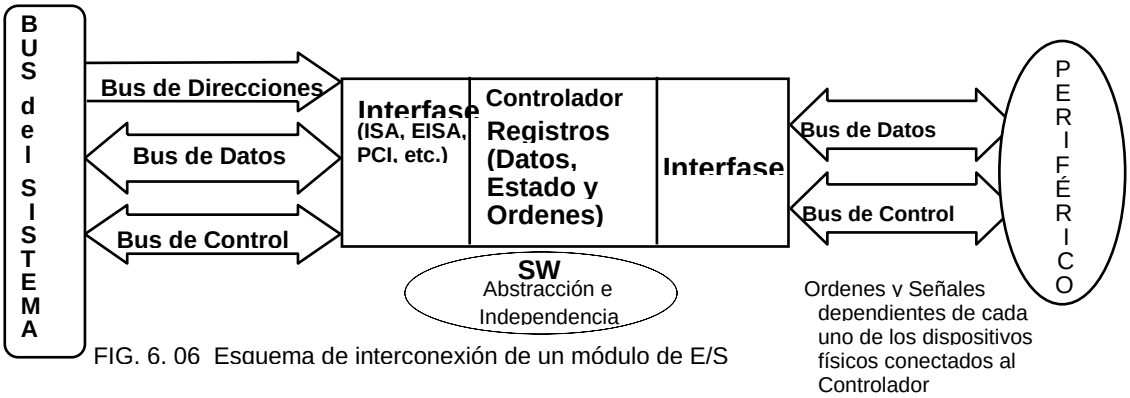


FIG. 6. 06 Esquema de interconexión de un módulo de E/S

Desde el punto de vista del S.O. interesa la segunda etapa.

La segunda etapa es la que produce la abstracción, desde el punto de vista del software, ya que presenta al dispositivo como un conjunto de registros dedicados (datos, órdenes y estados) que se leen o escriben en cada operación de E/S. A éstos registros generalmente se los denominan ***puertos de E/S (Ports).***

Los puertos pueden ser series, paralelos, controladores de discos flexibles, de discos duros, de fax - módem, de redes locales, de graficadores, de juegos, temporizadores programables (PIT Programmable Interval Timer), etc. como se observa en la Fig. 6.06, que generalmente se integran en un solo chip como el que se presenta en la Figura 6.07.

De acuerdo a la transferencia de datos de E/S, un puerto puede ser unidireccional o bidireccional en función a los dispositivos que se interconectan (ejemplo: Impresora unidireccional, disco bidireccional, etc.).

Los Registros Buffers se utilizan para el almacenamiento temporal de datos hasta que se pueda completar la transferencia de los mismos.

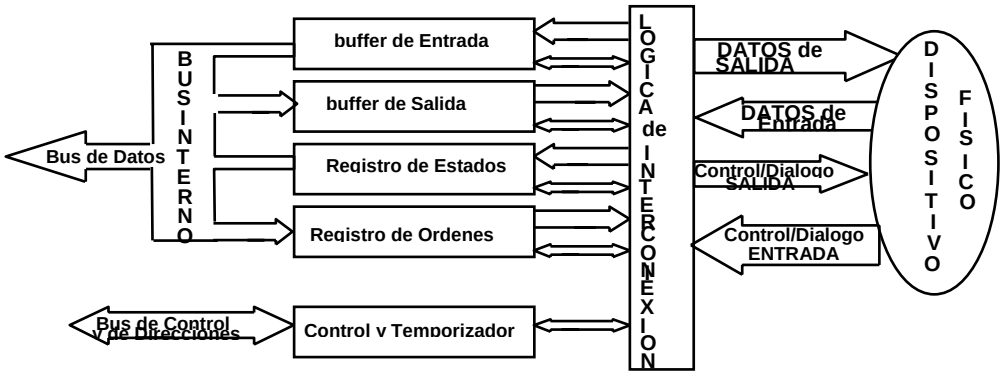


FIG. 6.07 Ejemplo de un controlador bidireccional.

Los registros de Estados y de Órdenes son utilizados por el mecanismo de diálogo para completar el protocolo de sincronización en el intercambio de datos. En particular los Registros de Estado pueden utilizarse para indicar la situación o condición en que se encuentra, ya sea dispositivos, controladores o canales y estos son:

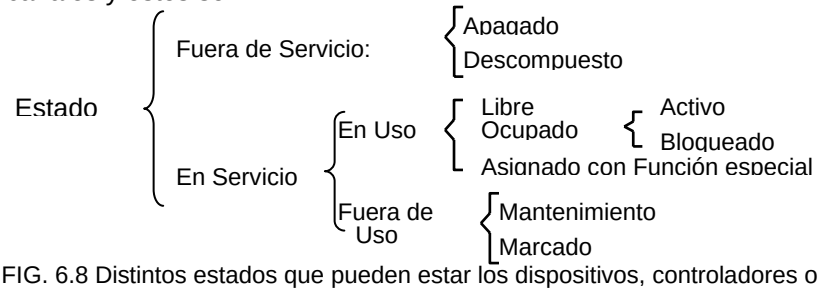


FIG. 6.8 Distintos estados que pueden estar los dispositivos, controladores o canales

Ejemplo de un Dispositivo asignado con función especial: El Disco del Sistema y la Consola. Marcado es cuando un operador separa un Dispositivo (Ejemplo Unidad de Cinta o una impresora con un dado formulario) “marcándolo” para que no pueda ser asignado a un nuevo trabajo por el S.O. cuando se libere su uso y pase al pool de recursos libres, si no a un determinado y específico trabajo que va a usar ese dispositivo en particular.

El controlador de bloques o de caracteres tiene como función principal, la conversión del flujo de bits, generados por el periférico, en un bloque de bytes cuando la operación es una Entrada o genera un flujo de bits cuando es una Salida.

Para ello verifica que no presenten errores en la lectura o escritura. Si detecta la presencia de errores intenta (varias veces) realizar la operación solicitada para corregir el problema. Si no tiene éxito, al cabo de esos intentos, entonces convoca al S.O. para un tratamiento del error en un nivel superior. Si la operación no presenta errores o fue exitosa la corrección en los reiterados intentos, recién transfiere los datos.

Un segundo objetivo, no menos importante que el anterior, es lograr que las operaciones de E/S sean eficientes y de un buen rendimiento.

Las Direcciones de los puertos pueden estar implementadas en dos formas;

1. En el espacio de direcciones de Memoria Central asignado a cada controlador (mapeados en memoria), por lo que los puertos pueden ser accedido y operado como si fuera la Memoria Central.
2. En un espacio de direcciones separados (mapeados en E/S o E/S aislada). Los accesos a los puertos deben ser realizados por medio de instrucciones hardware especiales que activan las señales de direccionamiento del espacio de E/S (ejemplo ent_port (in) o sal_port (out)).

La primera es más eficiente, mientras la segunda requiere efectuar copias de direcciones en los registros internos del procesador (acción que recarga el procesamiento de E/S pero es más simple de implementar y no ocupa espacio de memoria central por lo que es un método muy utilizado). Ampliaremos estos conceptos en el punto 6.5 de éste módulo

En general el uso de los Registros de Estados es sólo de lectura y el contenido de la información es una colección de bits posicionales en que cada posición indica una condición. Esta condición se verifica si se cumple (Verdadera o Falsa) y es utilizada por el S.O..

Los Registros de Ordenes son los que indican que función deberá realizar el módulo de E/S. Las órdenes provienen del tipo de servicio solicitado por el proceso, son enviados desde el procesador al controlador y a nivel de Software desde el S.O. al Driver del dispositivo.

Generalmente existen dos tipos de órdenes:

1. Las de inicialización o arranque (durante el proceso de bootstrapping o arranque en frío) del dispositivo, en que se prefija su modo de funcionamiento particular, su protocolo de comunicaciones y diálogos, mecanismos de transferencia de datos, generación de tablas, etc.
2. Las de uso normal generadas por todas las operaciones, temporizaciones, transferencia de datos, controles, interrupciones y manejos de errores.

Los registros de órdenes, generalmente, son sólo de escritura y modifican algunos bits del registro, mientras que el resto permanece sin alterar.

Como ejemplo un controlador de disco contiene la electrónica para el intercambio de datos, señales de control y de estado con el módulo de E/S más la electrónica para controlar a los mecanismos de lectura y/o escritura del disco. En un disco de cabezas fijas, el transductor es capaz de convertir patrones magnéticos en la superficie móvil del disco y los bits en el buffer del dispositivo (Figura 6.02 y 6.10). Un disco con cabezas móviles debe ser también capaz de provocar que el brazo del disco se mueva radialmente hacia adentro y hacia afuera de la de la superficie del disco.

6.4.3. Procesadores de E/S (IOP)

En los grandes equipos (Mainframe) generalmente se conectan mediante varios buses y a un procesador especial de E/S que se ocupa de toda la gestión, o un canal. A ésta configuración se denomina subsistema de E/S que hace de interfase con el procesador y memoria central. Todas las operaciones sobre periféricos se arrancan con una orden generada en el procesador para el procesador de E/S o canal, quien realizará las transacciones en forma independiente asumiendo el completo control de las mismas.

El **IOP** (Input-Output Processor) tiene acceso directo a memoria central y contiene una cantidad de canales independientes de E/S de datos sin intervención de la CPU. Los canales proveen las vías de comunicación (***path***) entre el IOP y las unidades de control (U.C.) de los dispositivos y los dispositivos que están físicamente conectados al canal. El sistema de bus de IOP comunica a las U. C. de E/S (Unidad de Control) con los subcanales (elemento pasivo sin capacidad de procesamiento lógico) del IOP mientras que el IOP se comunica con el bus del sistema o el bus de memoria central. Los canales de E/S pueden existir solos sin un IOP como pequeños procesadores que hacen uso de DMA (Direct Memory Access) para pocos dispositivos.

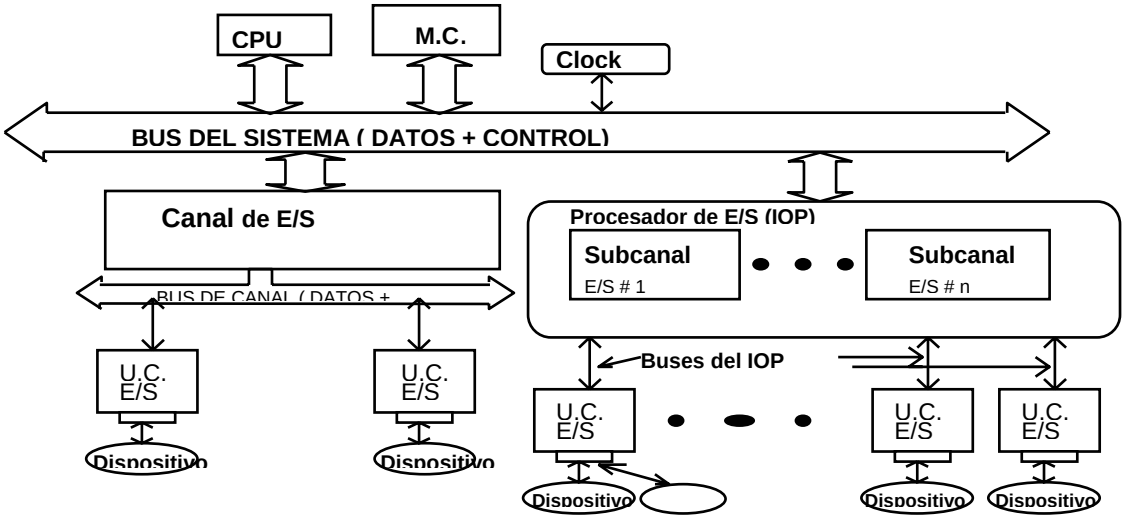


Figura 6.9 Esquema de E/S con Procesadores de E/S.

Para realizar una transacción de E/S el procesador genera una señal de arranque y la dirección del dispositivo al canal del cual depende el dispositivo en cuestión. Se verifica si la ruta está libre y el canal obtiene la dirección de comienzo del programa de canal para ejecutarlo. Si el path no está libre encola la transacción informando al procesador de la situación.

Completaremos este punto con la descripción del Sistema utilizados por IBM mas adelante en este modulo

6.4.4. Dispositivos Externos

Un sistema de computación no tiene ninguna utilidad sin algún medio para el ingreso o egreso de datos. Las operaciones de E/S se llevan a cabo a través de un amplio conjunto de dispositivos externos que proveen los medios para intercambiar datos entre el ambiente externo y el computador. Un dispositivo externo se acopla al computador a través de una conexión con un módulo de E/S (Figura 6.01). Esta conexión es usada para intercambiar señales de control, de estados y datos entre el módulo de E/S y el dispositivo externo. Un dispositivo externo conectado a un módulo de E/S normalmente es referido como un *dispositivo periférico.*

Según Stallings se puede clasificar a grandes rasgos a los dispositivos externos en tres categorías:

* ***legibles por el ser humano***: adecuados para la comunicación entre el hombre y la máquina
* ***legibles por máquinas***: adecuados para la comunicación con el equipamiento.
* ***comunicacionales*****:** adecuados para la comunicación con dispositivos remotos.

Ejemplos de los dispositivos legibles por el ser humano son las terminales de visualización por vídeo (VDT - Vídeo Display Terminal) y las impresoras. Ejemplos de dispositivos legibles por máquinas son los discos magnéticos, sistemas de cintas, sensores y actuadores, como los utilizados en aplicaciones de robótica. Obsérvese que se considera a los discos y las cintas magnéticas como dispositivos de E/S, mientras que pueden ser presentados como dispositivos de memoria para el almacenamiento de datos.

Desde un punto de vista funcional, estos dispositivos son parte de una jerarquía de memoria. Desde un punto de vista estructural, estos dispositivos son controlados por módulos de E/S. Los dispositivos de comunicación permiten al computador intercambiar datos con un dispositivo remoto, el cual puede ser legible por el ser humano, como ser una terminal, o ser legible por la máquina, o aún otro computador.

En términos muy generales, la naturaleza de un dispositivo externo es indicada en la Figura 6.10. La interfase con el módulo de E/S se establece mediante señales de control, de Estado (status) y de datos. Los ***datos*** presentan la forma de un conjunto de bits a ser enviados o recibidos desde el módulo de E/S. Las ***señales de control*** determinan la función que el dispositivo deberá ejecutar, como ser, enviar datos al módulo de E/S (GET, INPUT o READ), aceptar datos del módulo de E/S (PUT, OUTPUT o WRITE), reportar el estado, o ejecutar alguna función de control particular al dispositivo (por ejemplo, posicionar la cabeza del disco). **Las *señales de status*** indican el estado del dispositivo. Ejemplos son READY / NOT READY para mostrar si el dispositivo está listo o no para la transferencia de datos.

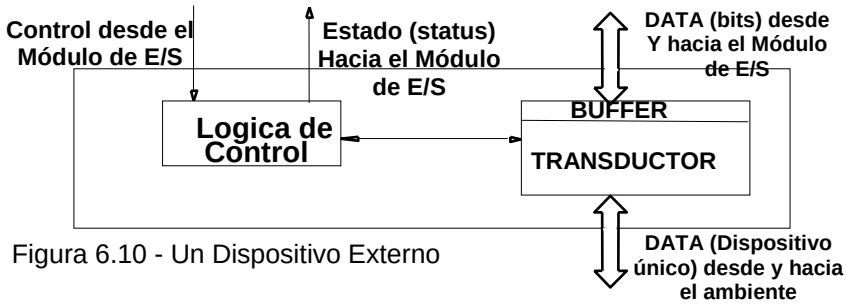


Figura 6.10 - Un Dispositivo Externo

La ***lógica de control*** asociada con el dispositivo controla la operación del dispositivo en respuesta a la directiva proveniente del módulo de E/S. El ***transductor*** convierte los datos en su naturaleza eléctrica a otras formas de energía (magnética, por ejemplo) durante un output, y de esta última a energía eléctrica (bits en memoria, flip-flops) durante el input. Típicamente, se asocia un buffer con el transductor para mantener temporariamente los datos a ser transferidos entre el módulo de E/S y el ambiente externo; un tamaño de buffer de 8 a 32 o más bits es algo muy común e incluso pueden tener varios KBy.

La interfase entre el módulo de E/S y el dispositivo externo será examinada posteriormente. La interfase entre el dispositivo externo y el ambiente está fuera del alcance de nuestra discusión, pero consideraremos algunos ejemplos.

Terminal de Visualización por Vídeo

Cada computadora tiene una o más terminales para comunicarse con ella. Existen muchas terminales diferentes, y una de las funciones del driver de terminales es ocultar esas diferencias.

Hardware de las Terminales

Desde el punto de vista del S.O., las terminales se dividen en dos categorías, basadas en como se comunican con el.

La primera categoría es la de las terminales que se comunican vía el standard RS-232. Consisten en un teclado y un display que se comunican usando una interfase serial de un bit por vez. Existen unos chips que convierten los caracteres en seriales y viceversa (se llaman Universal Asynchronous Receiver - Transmitters-UARTs)

La segunda son las TERMINALES CON MAPEO DE MEMORIA. No se comunican con la computadora a través de una línea serial. Son una parte integral de la computadora misma. Estas terminales se interfasean con una memoria especial llamada VÍDEO RAM, que forma parte del espacio de direccionamiento de la computadora y se direcciona al igual que la memoria común. En la vídeo RAM también hay un chip llamado vídeo controller. Este chip saca bytes de la vídeo RAM y genera señales que se envían al monitor. La señal del controlador de vídeo modula el rayo de electrones del Tubo de Rayos Catódicos (TRC o CRT Catode Ray Tube).

Las terminales de BIT-MAPS usan el mismo principio, excepto que cada bit en el vídeo RAM controla directamente un pixel en la pantalla.

Decíamos que el medio más común para un usuario para comunicarse con un computador es la terminal de vídeo (VDT). Una terminal consiste de un teclado y de una unidad de visualización, o ***pantalla***. El usuario provee el input a través de un teclado. Este input es transmitido al computador y puede ser también visualizado en la pantalla, de manera que el usuario puede ver qué es lo que está transmitiendo. Adicionalmente, la pantalla visualiza los datos enviados por el computador.

La unidad elemental de intercambio es el ***caracter*****.** Los caracteres son de dos tipos: visualizables o de control. Los caracteres visualizables son los caracteres alfabéticos, numéricos y signos especiales que el usuario puede ingresar con el teclado y que pueden ser observados en la pantalla. Los caracteres de control son interpretados para causar una acción específica al dispositivo, como ser el retorno-de-carro.

Hay un código asociado a cada caracter para poder representarlo en forma binaria. Los códigos típicos tienen una longitud de 5 a 7 bits. La Tabla 6.2 muestra los códigos para los caracteres visibles y los de control usados en el standard ASCII (American Standard Code for Information Interchange) cuya longitud es de 7 bits.

Durante una operación de output, estos códigos son transmitidos al dispositivo externo desde el módulo de E/S. El transductor interpreta éste código y envía a la unidad de visualización las señales electrónicas requeridas para ejecutar la función de control solicitada. Durante una operación de input, cuando una tecla es oprimida por el usuario, se genera una señal electrónica, la cual es interpretada por el transductor y traducida en un patrón de bits según el correspondiente código ASCII (u otro). En el computador este texto puede ser almacenado en este mismo código (u otro).

			POSICIONES DE BIT 7, 6, 5							
			000	001	010	011	100	101	110	111
			0	1	2	3	4	5	6	7
POSICIONES DE BITS 4, 3, 2, 1	0000	0	NUL	DLE	SP	0	@	P	´	p
	0001	1	SOH	DC1	!	1	A	Q	a	q
	0010	2	STX	DC2	"	2	B	R	b	r
	0011	3	ETX	DC3	#	3	C	S	c	s
	0100	4	EOT	DC4	$	4	D	T	d	t
	0101	5	ENQ	NAK	%	5	E	U	e	u
	0110	6	ACK	SYNC	&	6	F	V	f	v
	0111	7	BEL	ETB	'	7	G	W	g	w
	1000	8	BS	CAN	(	8	H	X	h	x
	1001	9	HT	EM	)	9	I	Y	i	y
	1010	10	LF	SUB	*	:	J	Z	j	z
	1011	11	VT	ESC	+	;	K	[	k	{
	1100	12	FF	FS	,	<	L	\	l	|
	1101	13	CR	GS	-	=	M	]	m	}
	1110	14	SO	RS	.	>	N	^	n	~
	1111	15	SI	US	/	?	O	_	o	DEL

TABLA 6.2 - a) Código ASCII

FORMATO DE CONTROL	
BS	(BackSpace) Retroceso: Indica el movimiento del mecanismo impresor o cursor de pantalla una posición hacia atras
HT	(Horizontal Tab) Tabulador Horizontal: Indica el movimiento del mecanismo de impresión o cursor de pantalla hacia adelante hasta el próximo "tabulador" preasignado o posición de detención.
LF	(Line Feed) Alimentador de Línea: Indica el movimiento del mecanismo impresor o cursor de pantalla a comenzar en la próxima línea.
VT	(Vertical Tab) Tabulador Vertical: Indica el movimiento del mecanismo impresor o cursor de pantalla hasta la próxima, de una serie, línea de impresión.
FF	(Form Feed): Alimentador de Hoja: Indica el movimiento del mecanismo impresor o cursor de pantalla a la posición de comienzo de la próxima hoja, forma, o pantalla.
CR	(Carriage Return) Retorno de Carro: Indica el movimiento del mecansimo impresor o cursor de pantalla a la posición de comienzo de la misma línea.

TABLA 6.2 - b) Código ASCII - Formatos de Control

CONTROL DE TRANSMISION	
SOH	(Start of Heading): Comienzo de Cabecera: Usado para indicar el comienzo de una cabecera la cual puede contener información del destinatario (dirección) o ruteo.
STX	(Start of Text) Comienzo de Texto: Usado para indicar el comienzo de un texto y también la terminación de la cabecera.
ETX	(End of Text) Fin de Texto:Usado para terminar el texto que fue iniciado con el código STX.
EOT	(End of Transmission) Fin de Transmisión: Indica la terminación de una transmisión la cual pudo haber incluído uno o más "textos" con sus cabeceras.
ENQ	(Enquiry) Requisición: Usado para requerir una respuesta de una estación remota. Puede ser utilizado como para preguntar ¿ QUIEN ES UD.? a una estación para que se identifique.
ACK	(Acknowledgement) Reconocimiento: Es un caracter transmitido por un dispositivo receptor como una respuesta afirmativa (aceptación) al transmisor. Se usa como una respuesta positiva a los mensajes de interrogación (polling).
NAK	(Negative Acknowledgemente) Reconocimiento Negativo: Es un caracter transmitido por el dispositivo receptor como una respuesta negativa al transmisor. Se usa como una respuesta negativa a los mensajes de interrogación (polling).
SYNC	(Synchronous/Idle) Sincrónico/Ocioso: Usado por un sistema de transmisión sincrónica para lograr la sincronización. Cuando no hay datos que enviar, el sistema de transmisión sincrónica puede enviar en forma contínua los caracteres SYNC.
ETB	(End of Transmission Block) Fin de transmisión de Bloque: Indica la terminación de un bloque de datos con propósitos de comunicación. Se utiliza para ablocar datos en donde la estructura del bloque no está necesariamente relacionada con el formato bajo proceso.

TABLA 6.2 - c) Código ASCII - Formatos de Control de Transmisión

SEPARADORES DE INFORMACION		
FS	(File Separator) Separador de Archivo	Los separadores de información se pueden utilizar de forma opcional excepto que su jerarquía debe ser de FS (el que más abarca) al US (el que menos abarca).
GS	Group Separator) Separador de Grupos	
RS	(Record Separator) Separador de Registros	
US	(United Separator) Separador de Unidad	

TABLA 6.2 - d) Código ASCII - Formatos de Separadores de Información

CÓDIGOS MISCELÁNEOS	
NUL	(Null) Nulo: Ningún caracter. Se utiliza como relleno en el tiempo o para rellenar espacio en una cinta cuando no hay datos.
BEL	(Bell) Campana: Se utiliza cuando hay necesidad de llamar la atención al operador. Puede usarse para controlar dispositivos de alarma o alerta.
SO	(Shift Out) Fuera de Código: Indica que las combinaciones de códigos que siguen en el mensaje deberán ser interpretadas fuera de conjunto de caracteres estándard hasta encontrar el código SI.
SI	(Shift In) Dentro de Código: Indica que las combinaciones de códigos que siguen deben interpretarse dentro del conjunto de caracteres estándard.
DEL	(Delete) Borrado: Se utiliza para destruir los caracteres no deseados (por ejemplo, en una tarjeta perforada, se perfora un agujero en cada posición (cada agujero se interpreta como un "1").
SP	(Space) Espacio: Es un caracter no imprimible utilizado para separar las palabras, o para mover el mecanismo de impresión o el cursor de pantalla en una posición.
DLE	(Data Link Escape) Escape del Enlace de Datos: Es un caracter que cambiará el significado de uno o más caracteres contiguos que siguen. Puede proveer control suplementario, o permitir el envío de caracteres que tengan cualquier patrón de bits.
DC1, DC2, DC3 y DC4	(Device Control) Control de Dispositivos: Son caracteres para cntrolar dispositivos subordinados o funciones específicas de las terminales.
CAN	(Cancel) Cancelación: Indica que los datos enviados precedentemente dentro de un mensaje o bloque deben ser descartados (usualmente porque el transmisor ha detectado un error).
EM	(End of Medium) Fin del Medio: Indica el fin físico de una tarjeta, cinta u otro medio, o el final de la porción o dimensión requerida del medio.
SUB	(Substitute) Substitución: Substituye un caracter al que se ha encontrado que era erróneo o no válido.
ESC	(Escape) Escape: Es un caracter por el cual se puede proveer una extensión del código, en el significado alternativo que tendrán un número determinado de caracteres que le siguen.

TABLA 6.2 - e) Código ASCII - Formatos de códigos miscelaneos (varios)

Discos

Tienen tres ventajas principales sobre la memoria central:

1. La capacidad de almacenamiento es mucho mayor
2. El precio por bit es mucho menor
3. No es un almacenamiento volátil

Los discos magnéticos consisten en una superficie circular recubierta de un material magnetizable. El disco gira bajo una cabeza lecto-grabadora y la cabeza se mueve en la dirección radial. A diferencia de los sistemas de cintas, en que las cabezas permanecen fijas, se mueven tanto el disco como la cabeza. De este modo, los discos permiten un acceso directo a los datos. Cada cara del disco está dividida en pistas concéntricas. Los discos prevén la posibilidad de que se grabe una porción de la pista que puede ser de tamaño fijo o variable. Cuando son de longitud fija o predeterminada por el fabricante se los llama sectores, y se dice que el disco está sectorizado. Si es variable se puede grabar cualquier cantidad de información. En los discos sectorizados cada pista está dividida en sectores. Un sector es la mínima unidad que se puede leer o grabar y generalmente es de 512 Bytes.

Hardware del Disco

Tiene la ventaja de que se pueden hacer búsquedas simultáneas en distintos dispositivos, esto se llama búsquedas superpuestas (overlap). Aunque no se puede leer o escribir simultáneamente. Básicamente el Hardware está compuesto por la Unidad de disco que se asocia con la controladora que puede manejar más de una Unidad.

El Soporte de información es un disco que pueden ser rígido o flexible.

Tipos de Discos: Disquette:

Con el advenimiento de la computadora personal, se necesitaba de un medio para la distribución del software. La solución se encontró en **el disquette** o disco flexible, un medio pequeño y removible, llamado así porque los primeros discos eran físicamente flexibles. El disquete fue en realidad inventado por la empresa IBM para guardar información sobre el mantenimiento de sus macrocomputadoras, para el personal de servicio, pero fue adoptado muy rápido por los fabricantes de computadoras personales como un medio conveniente para distribuir el software en venta. Los discos flexibles consisten de un disco de plástico dentro de un sobre cuadrado.

Dentro del sobre se encuentra un material no abrasivo que facilita el movimiento del disquete. Dicho material no frena ni desgasta la superficie magnética a velocidades bajas, pero tampoco permite aumentar la frecuencia de giro (alrededor de 300 r.p.m. revoluciones por minuto). Aún con un rozamiento pequeño, el

calor generado podría deformar el material o alterar el patrón magnético. Como consecuencia de esto la velocidad de rotación está limitada. El sobre tiene una ranura por donde la cabeza lecto-grabadora se apoya sobre el disco y un orificio índice que indica el comienzo de la pista. Los discos flexibles son sectorizados. También disponen de una ranura que los protege contra la escritura.

Los disquetes de 3.5 pulgadas vienen en un empaque rígido para su protección, por lo que no son precisamente flexibles. Además, como almacenan mas información y están mejor protegidos, y dado que se están investigando nuevos materiales de grabación que permitirían densidades del orden de 130 MBy de datos, con el tiempo reemplazarán a los de 3.5 pulgadas.

Por otra parte, en la tecnología de discos flexibles, la cabeza toca la superficie magnética; la flexibilidad del medio impide que movimientos bruscos del cabezal deterioren la plancha plástica en forma irreversible. Como resultado, tanto las superficies como las cabezas se desgastan más rápido. Para reducir el desgaste, las computadoras personales retraen las cabezas y detienen la rotación cuando el dispositivo no lee o escribe. En consecuencia, cuando se da el siguiente comando de lectura o escritura, existe un tiempo de espera de alrededor de medio segundo mientras el motor alcanza la velocidad requerida.

Discos Ópticos (CD-ROM y DVD):

Otro dispositivo que tecnológicamente se está haciendo popular son los **Discos Ópticos.**

Estos discos se preparan usando un láser de alto poder para perforar agujeros de un micrón (10^{-6} partes de un metro) en un disco matriz; después se hace un molde que se usa para imprimir copias en discos plásticos, del mismo modo en el que se elaboran los discos de pasta o vinilo. Luego se aplica en la superficie una delgada capa de aluminio, seguida de otra de plástico transparente para protección. Los discos CD ROM se leen con dispositivos similares a los de los reproductores de discos compactos de audio, por medio de un detector que mide la energía reflejada de la superficie al apuntar a esta un láser de bajo poder. Los agujeros, que se denominan huecos, y las áreas sin perforar entre estos, que llamaremos superficie plana, tienen diferente reflectividad, lo que hace posible distinguir entre ambos.

Esta tecnología tiene algunas consecuencias importantes: en primer lugar como los discos son impresos en lugar de grabados como los discos flexibles convencionales, pueden producirse en masa a muy bajo precio con dispositivos por completo automatizados. Por otra parte, ya que la impresión de discos de aluminio con cubierta plástica no es muy precisa, la información digital contiene por lo general muchos errores.

Este problema de los errores puede atacarse en dos formas. Primero, la cabeza lectora de la unidad contiene un espejo de precisión manejado por un servomecanismo que se utiliza para rastrear la superficie y compensar las imperfecciones de fabricación. Y segundo, los datos se graban usando un método complejo denominado código de corrección de errores de Reed Solomon. Este método de codificación usa más bits que el código de Hamming, pero puede corregir errores múltiples.

Aun mas, en lugar de usar los huecos para los ceros y las superficies planas para los unos (o viceversa), cada transición hueco-superficie plana o superficie plana-hueco representa un bit de valor 1 y el intervalo entre dos transiciones indica el número de ceros que están presentes entre los bits con valores de 1. Los datos se graban en grupos de 24 bytes, cada uno de los cuales se extiende de 8 a 14 bits usando el código de Reed-Solomon. Entre cada grupo, se agregan 3 bits especiales y se incorpora un byte de sincronización para conformar un cuadro. Un grupo de 98 cuadros forma un bloque que contiene 2K bytes de datos del usuario y es la unidad básica direccionable. Aunque este esquema es complicado y desperdicia un área considerable del disco, proporciona una confiabilidad en extremo elevada utilizando un medio de bajo costo.

Otro sistema utilizado es el indicado en la Figura 6.11-c.

La información en los discos CD ROM se graba en una sola espiral continua, a diferencia de los cilindros y pistas de los discos magnéticos.

Potencialmente, los discos CD ROM son útiles para la distribución de grandes bases de datos, en especial aquellas que combinan textos e imágenes. Otra aplicación de estos discos podría ser un curso de instrucción asistido por computadora, consistente en miles de diapositivas en color, cada una acompañada de una narración de audio de diez segundos.

Discos Ópticos (WORM):

No obstante este enorme potencial, no puede escribirse en discos CD ROM, lo que limita su utilidad como dispositivo de almacenamiento para computadora. El deseo de obtener un medio en el que se pudiera escribir, llevo a la siguiente etapa, el **disco óptico WORM** (de una sola escritura y lecturas múltiples). Este dispositivo permite al usuario escribir el mismo sobre el disco. Sin embargo, una vez que se ha perforado un hueco en la superficie, este ya no puede borrarse.

Aunque no están adaptados para crear y borrar archivos temporales de trabajo, dada su gran capacidad, se puede pensar en ir agregando uno tras otros archivos temporales hasta que se llene el disco y después tirarlo.

Es claro que la existencia de discos de una sola escritura tiene un gran impacto en la forma de desarrollar el software. Al no ser posible modificar los archivos, se tiende a sugerir un sistema diferente, en el cual un archivo es en realidad una secuencia de versiones inmutables, ninguna de las cuales puede alterarse y en donde cada una reemplaza a la anterior. Este modelo difiere mucho del utilizado en los discos magnéticos de actualización in situ.

Otros Discos Ópticos:

Un medio óptico borrable es la tercera etapa en la evolución de los discos ópticos, la cual utiliza una tecnología magnética. El disco plástico es recubierto con una aleación de metales tan extraños como el terbium y el gadolinium. Estos metales tienen la interesante propiedad de que a bajas temperaturas son insensibles a los campos magnéticos pero en altas temperaturas su estructura molecular se alinea con cualquier campo magnético presente.

Para hacer uso de esta propiedad, las cabezas de la unidad tienen un láser y un magneto. El láser dispara al metal una ráfaga de luz ultracorta, elevando su temperatura pero sin horadar la superficie. Al mismo tiempo el magneto esta emitiendo un campo en una de sus dos direcciones. Al terminar el impulso del láser, el metal ha quedado magnetizado en una de dos posibles direcciones, representando un 1 o un 0. Esta información puede volverse a leer del mismo modo que en un disco, usando un láser más débil que no caliente la superficie. El disco también se puede borrar y reescribir al igual que en la primera ocasión.

No es probable que los discos ópticos grabables sustituyan a los convencionales discos rígidos por algún tiempo -posiblemente muchos años-, por dos razones principales: primero, su tiempo de búsqueda es un orden de magnitud mas lento que en los discos rígidos y, segundo, su velocidad de lectura de datos es también un orden de magnitud menor. Comparados, el desempeño general de los discos magnéticos es sencillamente mucho mejor. Mientras que los discos ópticos sin duda alguna mejoraran sus tiempos, los magnéticos probablemente mejoraran en la misma medida para mantener su ventaja. No obstante, para aquellas aplicaciones en las que disponer de una gran cantidad de almacenamiento removible es vital, los discos ópticos tienen un gran futuro.

Discos Rígidos (Hard Disk – HD):

Los **discos rígidos** pueden ser fijos o removibles. Los fijos forman una misma cosa con la unidad, en cambio en los removibles se pueden cambiar los paquetes de discos. En general los discos fijos son más veloces y más confiables para mantener la información, pero la capacidad es limitada. Los removibles son más lentos, y como se mueven más están sujetos a errores o problemas.

Los discos rígidos están constituidos por varios platos de un material duro, recubierto con una sustancia similar a la que cubre los disquetes. Los platos, apilados paralelamente en un eje giratorio que los atraviesa por el centro, están separados lo suficiente como para permitir la introducción de la cabeza lecto-grabadora. Normalmente la primera y última cara no se utiliza en los removibles. La dureza del material que forma los platos disminuye los riesgos de deformaciones, y aumenta la precisión del sistema. Adicionalmente, la ausencia de rozamiento con materiales sólidos (los platos quedan sostenidos únicamente por el eje central) permite un aumento de velocidad de rotación hasta los límites que imponen otros aspectos de la tecnología (la electrónica de la codificación, por ejemplo, o la resistencia estructural del material). La introducción de partículas de polvo entre el disco y la cabeza originan fallas. Es por esto que los discos rígidos se encuentran encerrados con el mecanismo de tracción y lecto-escritura en una carcaza semihermética, libre de polvo.

Hay dos materiales magnéticos principales con los que se recubren las superficies de los platos. El óxido de hierro, similar al empleado en las cintas de audio, es uno de los más usados y económicos. Tanto los discos rígidos como los flexibles hacen uso de esta tecnología. El otro material es una aleación metálica inoxidable. El proceso de fabricación de esta superficie es más caro, pero permite una mayor densidad de grabación.

En los discos rígidos los movimientos son más precisos. Esto permite almacenar una mayor densidad de información.

Si bien las cabezas de los discos rígidos no tocan la superficie del disco, se encuentran lo más cercanamente posible para asegurar buenas señales (tanto en la grabación como en la lectura). Como las cabezas no tocan la superficie del disco rígido, este se encuentra rotando continuamente, ahorrándose el tiempo necesario para que el disco alcance la velocidad correcta.

Como dijimos anteriormente el disco se divide en caras, las caras en pistas y las pistas en sectores. Todas las pistas que en un disco rígido pueden ser direccionadas sin que las cabezas se muevan (las pistas equivalentes sobre distintas superficies) constituye un cilindro. Para acceder a un sector, debemos especificar la superficie, la pista, y el sector. Para ayudar al dispositivo (drive) de disco a localizar la posición de un sector, el drive graba marcas de sector entre los sectores.

La cabeza lecto-grabadora se mueve a la pista correcta (tiempo de búsqueda) y electrónicamente se selecciona la cara correcta; entonces, esperamos (tiempo de espera) que el sector requerido pase bajo la

cabeza. El tiempo de búsqueda depende del tiempo que les tome a las cabezas del disco moverse, cuanto más alejados están las pistas, mayor ser el tiempo de búsqueda. En las unidades de discos de varios platos todas las cabezas se mueven solidariamente, pero para transferir información lo hacen una sola por vez. Como el cambio de cabeza es electrónico y muy rápido, si toda la información está grabada en un mismo cilindro se tardará menos, para leerla, que si está en distintos cilindros. Porque el tiempo de posicionamiento de la cabeza sobre el cilindro, al ser mecánico, es muy superior.

La información en los discos se graba en serie a lo largo de la pista y no en paralelo como en las cintas magnéticas. Cada pista del disco es lo mismo que un canal de grabación de una cinta.

Las cabezas lecto-grabadoras pueden ser incorporadas, si forman parte del paquete de discos, o externa, si forma parte de la unidad de disco. Las cabezas también pueden ser fijas o móviles. Una unidad con cabezas fijas tendrá una cabeza por cada pista. Por lo general cabezas fijas implican discos fijos. Un paquete puede tener algunas cabezas fijas y otras móviles a la vez. Por ejemplo una cabeza fija sobre la pista en que se encuentra el directorio.

Si el disco no está sectorizado hay una marca fija que indica el comienzo de una pista. Pueden existir platos donde hay marcas lógicas para posicionar las cabezas en el inicio de cada pista. A partir del dato, se graba información de control: un primer registro, el home address, le dice al mecanismo de grabación, en qué pista y cilindro está, si la pista es útil, la cantidad de bytes libres. Hay un home address por cada pista de cada cilindro. Por cada registro de datos de cada pista hay un campo cuenta. Este campo contiene el número de registro y la longitud del área de datos.

Los sectores no están numerados en forma sucesiva, para que actúen como “gaps”, en el caso en que se necesiten leer dos sectores consecutivos en numeración, después de leer el primero. Esto se conoce como **interleave**. Mientras éste se procesa, se puede leer el segundo, lo que posibilita disminuir tiempos de acceso.

Un concepto relacionado con el hardware de E/S es el ***interleaving*** (factor de separación o salteado). Este concepto está relacionado con los discos, con el hecho de que la información de un sector de disco se lea sobre un buffer del controlador de disco antes de transferirlo a memoria, con el hecho de que el disco no deja de girar y con el hecho de que realizar el checksum y transferir a memoria el sector conlleva un tiempo. Si se desea leer varios sectores lógicamente consecutivos (que estén ubicados físicamente consecutivos), en primer lugar se llevará el primer sector al buffer del controlador, se realizará el checksum y se transferirá a memoria. Como el disco no deja de girar, cuando el controlador solicite el siguiente sector, habrá que esperar a que éste de una vuelta casi entera para que las cabezas de Lecto-Escritora se sitúen al principio del segundo sector. Para evitar esta situación, los sectores lógicamente consecutivos no siempre se sitúan físicamente consecutivos.

Interleaving óptimo: aquel que permite leer/escribir todos los sectores de una pista (del 0 al último sector lógico de la pista) en el menor número de vueltas del disco.

En cada pista se graba información para llevar el control de paridad. No se graban 9 bits por byte como en las cintas. Se graban 8 bits y el procesador va acumulando cuantos bits de paridad hubieran correspondido. Luego de grabar el área de datos se graban los dos bytes con la cantidad acumulada de bits de paridad. Cuando se realiza la lectura se vuelve generar un bit de paridad por cada byte y se acumula del mismo modo que cuando se grabó (ver figura 6.11a) Por último se compara el valor grabado en el disco con el calculado al realizar la lectura.

Para organizar la información dentro del disco el sistema operativo lleva una tabla, la **VTOC (Volume Table Of Contents** que será más detallada y ampliada en el módulo 7), con información sobre todos los archivos del disco. Esta tabla está siempre ubicada a partir de una posición fija del disco y contiene un registro por cada archivo que está sobre el volumen. En la tabla se indica el nombre del archivo, cuanto ocupa, su posición sobre el volumen, etc..

Manejo de Disco

Existen diferentes aspectos del manejo del disco, del cual es responsable el Sistema Operativo.

Formateo del Disco:

Cuando se fabrica un disco, es esencialmente, un espacio en blanco, como un soporte de un medio sin un orden preestablecido, magnéticamente variable.

Antes de que el computador pueda usar el disco, éste debe ser dividido en pistas y sectores, que el computador pueda entender. Este proceso se llama formateo físico o de bajo nivel. Una vez formateado, el disco estará formado por un conjunto de sectores y pistas, que la cabeza lectora, grabadora puede direccionar.

Cada sector, tiene una cabecera (header), que contiene el número de sector y un espacio para un código de corrección de error (ECC) o un **código de detección de errores** (CRC Check Redundance Code) como se indica en la figura 6.11 a. Luego hay un campo de datos que también tiene área de código para detectar errores.

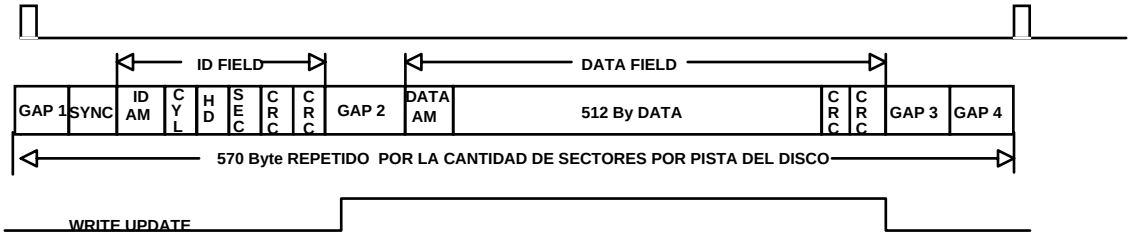


Fig. 6.11 a: FORMATO DE UN SECTOR DE UN DISCO TIPO WINCHESTER

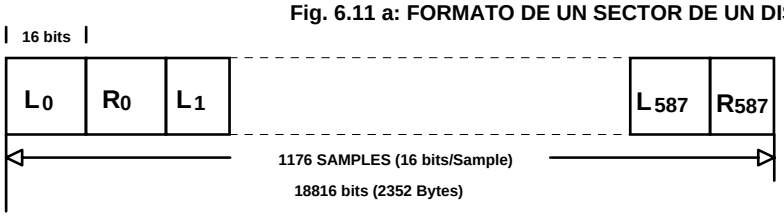


Fig. 6.11 b: FORMATO DE UN CD-(Compact Disc)- de MUSICA

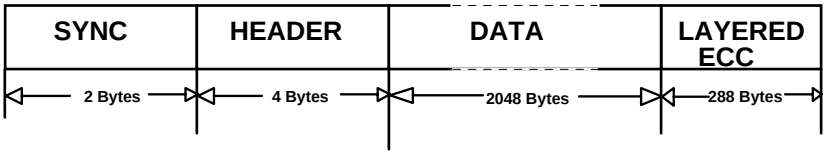


Fig. 6.11 c: FORMATO DE UN CD-ROM de DATOS

Este código se actualiza al momento de leer o grabar en cada sector y se compara con los valores leídos o grabados, respectivamente.

Algunos discos, vienen formateados de fábrica.

El sistema operativo necesita formatear lógicamente el disco, antes de usarlo. Este formateo lógico, escribe blancos (espacios) o cualquier otra información que necesite el Sistema Operativo para rastrear los contenidos del disco en el área de datos. Esto también se llama formateo en alto nivel.

Bloque de Inicio (Boot Sector)

Todo computador necesita un programa inicial para ejecutar. Este programa inicial, llamado bootstrap program, tiende a ser sencillo: inicializa todos los aspectos del sistema, desde los registros de CPU, hasta los controladores de dispositivos, o los contenidos de la memoria.

El bootrstrap program debe conocer como cargar el Sistema Operativo, y comenzar su ejecución. Para cumplir con este propósito, debe localizar el Kernel o núcleo del Sistema Operativo, cargarlo en la memoria y hacer un salto o desplazamiento hacia las direcciones iniciales.

La pregunta que surge es la siguiente: ¿Cómo hace el computador para conocer la ubicación del bootstrap program y comenzar la ejecución? El sistema, almacena una pequeña parte del bootstrap en memoria ROM, y el resto, en bloques de inicio, en una posición fija, en el disco. Este disco se conoce como disco de booteo o disco del sistema, y el sistema no puede funcionar sin él. El código almacenado en ROM solicita la transferencia de bloque, desde el controlador del disco, carga los datos de los bloques de inicio, en la memoria y comienza a ejecutar el nuevo código. Los cambios en el programa de booteo se hacen al reescribir los bloques de inicio.

Bloques Malos

Los discos son propensos a fallas. A veces la falla es total y entonces, el disco debe ser reemplazado por otro y sus contenidos recuperados desde copias de seguridad. En otras oportunidades, mas de un bloque de inicio se tornan ilegibles e ingrabables. Algunos discos ya vienen de fábrica con estas fallas, o bloques malos. Dependiendo del disco y de su controlador, estos bloques pueden ser manejados de varias maneras:

Para la familia de PC IBM, los bloques malos se manejan manualmente. El comando format busca los bloques malos, cuando el disco se formatea lógicamente y graba un valor especial en la FAT (File Allocation Table) para llamar a las rutinas de asignación, para que no usen esos bloques. Los datos que residen en los bloques malos, generalmente, se pierden.

Para cada sector malo, el controlador puede reemplazarlo lógicamente, con un sector de su spool.

Ejemplo: Cómo se desarrollan los pasos:

a) El Sistemas Operativo solicita sector 5, pista 20.
b) El controlador calcula los ECCs (Códigos de corrección de errores) o los CRC, y encuentra que ese sector esta malo, comunicándolo al Sistema Operativo.
c) La próxima vez que el sistema es booteado, se ejecuta un comando especial para avisar al controlador (IDE o SCSI) que reemplace el sector malo, por uno bueno.

d) La próxima vez que el sistema solicite el sector 5, pista 20, el pedido será trasladado por el controlador, a la dirección del sector reemplazante (por ejemplo sector 27, pista 153)

Manejo del Espacio de Swapping (Uso del espacio de Swapping)

El uso del espacio del swapping, es diferente, según el Sistema Operativo, dependiendo de los algoritmos de administración de memoria. Los sistemas que usan paginación, por ejemplo, simplemente almacenan páginas que fueron apiladas fuera de la memoria central.

La cantidad de espacio de swapping que necesita un sistema depende de la cantidad de memoria disponible, y de la forma en que esta es usada. En las computadoras personales, se necesitan unos pocos megabytes de disco.

Algunos Sistemas Operativos, como UNIX, permiten el uso de espacios múltiples de swapping. Estos espacios están dispuestos en discos separados.

Ubicación del Espacio de Swapping

El espacio de swapping, puede residir en dos lugares. puede ser colocado fuera del sistema de archivos, o puede ser una partición separada del disco. Si el espacio de swapping es un gran archivo, fuera del sistema de archivos, sus rutinas pueden ser usadas para crearlo, destruirlo, renombrarlo, etc. Aunque este método es sencillo de aplicar, es ineficiente.

La performance se puede incrementar al usar memorias caché para contener la información del bloque de locación, y usando herramientas especiales para asignar bloques secuenciales para archivos, pero el costo de esta estructura es alto.

Más comúnmente, el espacio de swapping es creado en una partición separada del disco. Los algoritmos de manejo de espacio de swapping se usan para asignar y desasignar los bloques. La ventaja de este método es la velocidad. La fragmentación interna aumenta, pero es aceptable porque los datos en el espacio de swapping generalmente existe por mucha menos cantidad de tiempo que los archivos en el sistema de archivos, y esos datos son accedidos mucho más frecuentemente que los que están en el sistema de archivos. Este método requiere que el espacio sea prefijado durante la creación del sistema de archivos, y esto sólo puede ser hecho destruyendo archivos que existen, o agregando nuevos discos. Esta es su mayor desventaja.

Parámetros de Rendimiento de Discos

Si bien se mostró en la fig. 6.03 todos los tiempos que intervienen en una E/S, los detalles reales de las operaciones de E/S con los discos dependen del computador, el sistema operativo y la naturaleza del canal de E/S y el hardware controlador de disco. En la figura 6.12 se muestra un típico diagrama de tiempos de la E/S a disco.

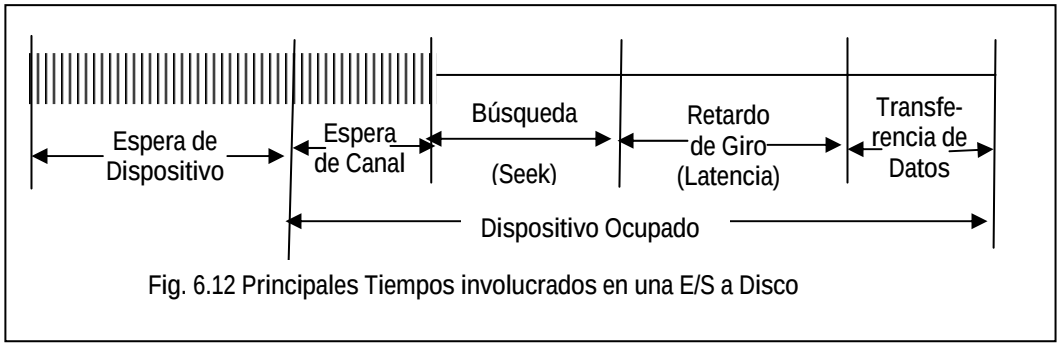


Fig. 6.12 Principales Tiempos involucrados en una E/S a Disco

Cuando la unidad de disco está operando, el disco gira a una velocidad constante. Para leer o escribir, la cabeza debe posicionarse en la pista deseada, al comienzo del sector pertinente. Si el sistema es de cabezas móviles, hay que mover la cabeza para elegir la pista. Si el sistema es de cabezas fijas, habrá que seleccionar electrónicamente una de ellas. En un sistema de cabezas móviles, el tiempo que se tarda en ubicar la cabeza en la pista se llama tiempo de búsqueda. En cualquier caso, una vez que se ha seleccionado la pista, el controlador del disco esperará hasta que el sector apropiado se alinee con la cabeza en su rotación. El tiempo que tarda el comienzo del sector en llegar hasta la cabeza se conoce como retardo de giro, o latencia de giro. La suma del tiempo de búsqueda y el retardo de giro es el tiempo de acceso, es decir, el tiempo que se tarda en llegar a la posición de lectura o escritura. Una vez que la cabeza está ubicada, se puede llevar a cabo la operación de Lectura o Escritura a medida que el sector se mueve bajo la cabeza; ésta es la parte de transferencia real de datos de la operación.

Además del tiempo de acceso y del tiempo de transferencia, en una operación de E/S intervienen algunos retardos. Cuando un proceso emite una petición de E/S, primero debe esperar en una cola a que el dispositivo esté disponible. En ese momento, el dispositivo queda asignado al proceso. Si el dispositivo

comparte un único canal de E/S o un conjunto de canales con otras unidades de disco, puede producirse una espera adicional hasta que el canal esté disponible. En ese punto se realizará la búsqueda con que comienza el acceso al disco.

En algunos sistemas grandes se emplea una técnica conocida como *detección posicional de giro (RPS[2]).* Esta técnica funciona como se explica seguidamente. Cuando se ejecuta la orden de búsqueda, se libera el canal para que pueda realizar otras operaciones de E/S. Cuando la búsqueda termine, el dispositivo debe averiguar el instante en que los datos van a pasar bajo la cabeza. A medida que el sector se aproxima a la cabeza, el dispositivo intenta restablecer la vía de comunicaciones con el computador central. Si la unidad de control o el canal están ocupados con otra operación de E/S, el intento de reconexión no tendrá éxito y el dispositivo debe dar una vuelta completa antes de intentar la reconexión, lo que se denomina *una falta de RPS.* Esta componente extra del retardo debe añadirse al diagrama de tiempos de la figura 6.12.

Tiempo de acceso (access time)

Termino frecuentemente usado en discusiones de desempeño, es el intervalo de tiempo entre el momento en que una unidad de disco (drive) recibe un requerimiento por datos, y el momento en que un drive empieza a despachar el dato. El tiempo de acceso de un Hard Disk es una combinación de tres factores: Tiempo de Búsqueda, Tiempo de Latencia y Tiempo de transferencia.

Tiempo de Búsqueda (Seek Time)

El tiempo de búsqueda es el tiempo necesario para mover el brazo del disco hasta la pista solicitada. Esta cantidad resulta difícil de obtener. Como la pista deseada puede estar localizada en el otro lado del disco o en una pista adyacente, el tiempo de búsqueda variara en cada búsqueda. En la actualidad, el tiempo promedio de búsqueda para cualquier búsqueda arbitraria es igual al tiempo requerido para mirar a través de la tercera parte de las pistas. Algunos fabricantes citan un tiempo de búsqueda pista a pista, el cual es simplemente la cantidad de tiempo para mover la cabeza de una pista a la pista adyacente. Los Discos Rígidos de la actualidad tienen tiempos de búsqueda pista a pista tan cortos como 2 milisegundos y tiempos promedios de búsqueda menores a 10 milisegundos y tiempo máximo de búsqueda (viaje completo entre la pista más interna y la más externa) cercano a 15 milisegundos.

El tiempo de búsqueda consta de dos componentes clave: el tiempo de arranque inicial del brazo y el tiempo que se tarda en recorrer los cilindros, una vez que el brazo haya tomado velocidad. Por desgracia, el tiempo de recorrido no es una función lineal con el número de pistas. Se puede aproximar el tiempo de búsqueda con la fórmula lineal:

$$T_s = m * n + s$$

donde
T = tiempo de búsqueda estimado
n = número de pistas recorridas
m= constante que depende de la unidad de disco
s = tiempo de arranque del brazo

Por ejemplo, un disco *Winchester* económico en un computador personal podría tener, aproximadamente, m = 0,3 ms y *s* = 20 ms, mientras que uno más grande y más caro podría tener *m* = 0,1 ms y *s* = 3 ms.

Retardo de Giro (Tiempo de latencia)

Cada pista en un Disco Rígido contiene múltiples sectores una vez que la cabeza de Lectura/Escritura encuentra la pista correcta, las cabezas permanecen en el lugar e inactivas hasta que el sector pasa por debajo de ellas. Este tiempo de espera se llama latencia. La latencia promedio es igual al tiempo que le toma al disco hacer media revolución y es igual en aquellas unidades que giran a la misma velocidad. Algunos de los modelos más rápidos de la actualidad tienen discos que giran a 18000 RPM reduciendo la latencia.

Los discos más populares, excepto los flexibles, giran normalmente entre 3600 a 7200 rpm (revoluciones por minutos), es decir, para 3600 rpm, una revolución tarda 16,7 ms. Por lo tanto, el retardo medio de giro será de 8,3 ms. para 7200 rpm vale la mitad. Los discos flexibles giran mucho más lentamente, generalmente entre 300 y 600 rpm. Por lo tanto, el retardo medio estará entre 100 y 200 ms.

Tiempo de Transferencia

El tiempo de transferencia con el disco depende de la velocidad de rotación de la forma siguiente:

[2] RPS significa Rotacional Position Search

$$T = b / (r N)$$

Donde:
T = tiempo de transferencia
b = número de bytes a transferir
N = número de bytes por pista
r = velocidad de rotación en revoluciones por segundo
Por tanto, el tiempo medio de acceso total puede expresarse como

$$T_a = T_s + (1 / 2 r) + (b / r N)$$

donde T_s es el tiempo medio de búsqueda.

TASA DE TRANSFERENCIA.

Los discos también son evaluados por su Tasa de transferencia, la cual generalmente se refiere a la Tasa en la cual los datos pueden ser leídos o escritos en la unidad. La velocidad de los discos, la densidad de los bits de datos y el tiempo de acceso afecta la Tasa de transferencia. La Tasa de transferencia es particularmente importante cuando se leen y escriben archivos grandes. Las unidades actuales tienen Tasas de transferencia que oscilan entre 5 y 30 megabytes/segundo.

La mayoría de los discos actuales incluyen una cantidad de memoria RAM que es usada como caché o almacenamiento temporal. Algunas especificaciones de disco se refieren a una Tasa de transferencia por ráfagas o la velocidad a la cual los datos pueden ser leídos o escritos en la caché.

Dado que los computadores y los discos se comunican por un bus de Entrada/Salida, la Tasa de transferencia actual entre ellos esta limitada por la máxima Tasa de transferencia del bus, la cual en la mayoría de los casos es mucho más lenta que la Tasa de transferencia del disco.

Algoritmos de Planificación del Brazo del Disco

Cuando la unidad de disco está operando, el disco gira a una velocidad constante. Para leer o escribir la cabeza debe posicionarse en la pista al comienzo del sector pertinente. Si el sistema es de cabezas móviles hay que mover la cabeza para elegir la pista, y si es de cabezas fijas habrá que seleccionar electrónicamente una de ellas. El tiempo para leer o escribir un bloque en el disco, decíamos que, básicamente está determinado por estos tres factores: el tiempo de búsqueda (movimiento del brazo al cilindro adecuado), la demora rotacional (hasta que pase el sector buscado por debajo de la cabeza) y el tiempo de transferencia además depende de la organización del archivo. El mas largo es el tiempo de búsqueda. Si los pedidos se aceptan de a uno y se realizan de a uno, no se puede hacer nada para optimizar el tiempo de búsqueda. Sin embargo, si mientras se realiza una operación se pueden registrar los pedidos surgidos es posible una mejora. Algunas unidades mantienen una tabla, ordenada por el número de cilindro, con todos los pedidos pendientes para cada cilindro juntos en una lista enlazada encabezada por las entradas a la tabla.

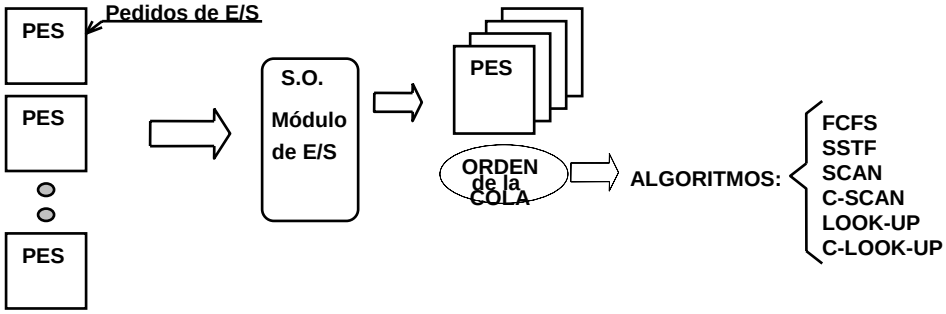


Figura 6.13 Ordenamiento de pedidos de lecturas o escrituras de disco

Existen varios algoritmos que esencialmente deben tener en cuenta el porcentaje de operaciones que se efectúan sobre el disco y se busca la mejor eficiencia o sea, el menor tiempo posible, ya que este condiciona el rendimiento de los procesos (Swapping, Paginación, Carga de programas, etc.) porque se produce una gran demora. Estos algoritmos se basan en **políticas de planificación**.

La manera más sencilla de planificación es **FIFO**, que tiene la ventaja de ser justa porque las peticiones son servidas en el orden de llegada. Sin embargo, esta política suele ser a menudo poco eficiente. Con un sistema **PRIridad** el control de la planificación queda aislado del control del software gestor del disco, no se persigue la optimización del uso del disco sino cumplir con otros objetivos del SO.

Otra política es la **LIFO**, que en sistemas de procesos de transacciones puede resultar provechosa. Sin embargo, esta política puede llevar a la inanición. Las tres políticas nombradas se basan en las propiedades de la cola o del proceso demandante, pero si la posición de la pista actual es conocida, puede emplearse una planificación en función del elemento demandado. La política de **SSTF** consiste en elegir la solicitud de E/S que requiera el menor movimiento posible del brazo del disco desde su posición actual, esta política puede generar inanición.

Otra política es **SCAN**, el brazo sólo se mueve en un sentido resolviendo todas las peticiones pendientes de su ruta hasta que alcance la última pista, entonces cambia la dirección de servicio y el rastreo sigue en sentido opuesto. Una mejora es la política **C-SCAN** que recorre el disco en una sola dirección, cuando se haya visitado la última pista el brazo vuelve al extremo opuesto del disco y comienza a recorrerlo de nuevo.

Con estas tres políticas mencionadas, es posible que el brazo no se mueva durante un tiempo considerable si uno o más procesos realizan muchos accesos a una misma pista. Para evitar esto se utiliza la política **SCAN de N** pasos o **FSCAN**. La primera divide la cola de peticiones en subcolas de longitud N y procesa las subcolas una a una mediante un SCAN. La política FSCAN usa dos subcolas, cuando comienza un rastreo todas las peticiones están en una de las colas y la otra está vacía y durante el recorrido las nuevas peticiones se colocarán en la segunda cola. Pero veamos en detalle a los siguientes algoritmos de planificación:

Planificador FCFS (First Come First Served o FIFO)

El más simple planificador de disco es, por supuesto, el planificador primero en llegar, primero en servirse (First Come First Served). Este algoritmo es fácil de programar y es intrínsecamente justo. Sin embargo, puede no proveer el mejor servicio (promedio). Consideremos, por ejemplo, una cola ordenada de disco con requerimientos o PES (Pedidos de Entrada / Salida) involucrando las siguientes pistas (tracks):

100, 200, 45, 120, 10, 130, 60, 70, 30 y 20; desde la primera listada (100) a la última (20).

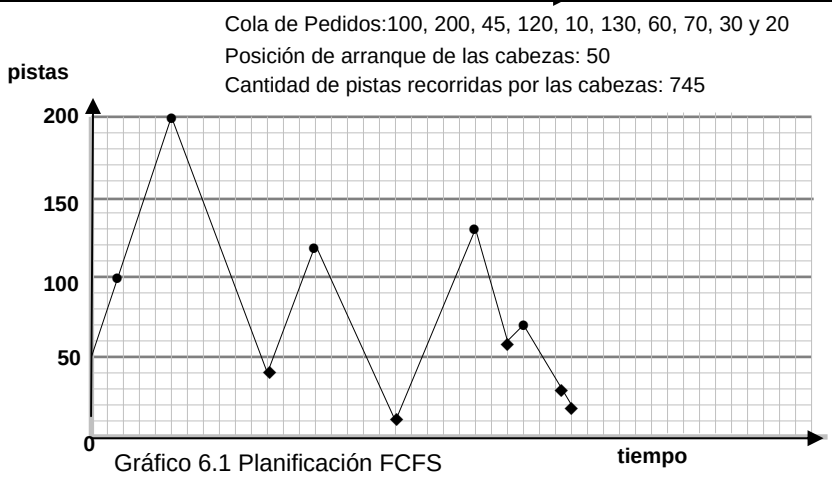


Gráfico 6.1 Planificación FCFS

Si la cabeza lecto-grabadora está inicialmente en la pista 50, por lo que primero se debe mover desde la 50 a la 100, luego a las 200, 45, 120, 10, 130, 60, 70, 30 y finalmente a la 20, para un movimiento total de la cabeza de 745 pistas (50 + 100 + 155 + 80 + 110 + 120 + 70 + 10 + 40 + 10 = 745).

El problema con esta planificación se presenta por el enorme movimiento del brazo de 120 a 10 y vuelta a la pista 130.

Si el requerimiento para tracks 45 y 10 podían ser servidos juntos, antes o después de los requerimientos 120 y 130, el movimiento total de la cabeza podría decrecer substancialmente y el tiempo promedio para servir cada requerimiento decrementaría, mejorando el rendimiento del disco.

Este algoritmo es muy útil para planificar discos con poca carga (pocos accesos) y es el más simple de programar, pero tiene mucho thrashing.

Planificador SSTF(Shortest Seek Time First)

Parece razonable servir juntos todos los requerimientos cercanos a la actual posición de la cabeza, antes de mover la cabeza para servir otro requerimiento lejano. Esta suposición es la base para el planificador de disco shortest-seek-time-first (SSTF).

El algoritmo SSTF selecciona el requerimiento con un mínimo tiempo de búsqueda desde la actual posición de la cabeza. Puesto que el tiempo de búsqueda es generalmente proporcional a la diferencia de pistas entre los requerimientos, implementaremos este método por movimiento de la cabeza a la pista más cercana en la cola de requerimientos.

Para nuestra cola de requerimientos del ejemplo propuesto, el pedido más cercano desde la posición inicial de la cabeza (50) es la pista 45. Una vez en la pista 45, el requerimiento más cercano siguiente es la pista 30. En este punto, la distancia a la pista 30 es 15 al igual que la 60 pero la cabeza está en movimiento hacia la menor, por lo que va a atender el pedido de la pista 30. Luego el requerimiento del track 20 lo considera más cercano y es el próximo servido. Continuando, sirve el requerimiento del track 10, luego 60, el 70, y luego el 100, 120, 130, y finalmente al 200.

Este método de planificación tiene un total de movimiento de cabeza de sólo 230 pistas - menos que un tercio la distancia necesitada para el planificador FCFS.

Este algoritmo resultaría en una importante mejora en el servicio de disco promedio.

Cola de Pedidos: 100, 200, 45, 120, 10, 130, 60, 70, 30 y 20
Cola de Atención de Pedidos: 45, 30, 20, 10, 60, 70, 100, 120, 130 y 200
Posición de arranque de las cabezas: 50
Cantidad de pistas recorridas por las cabezas: 230

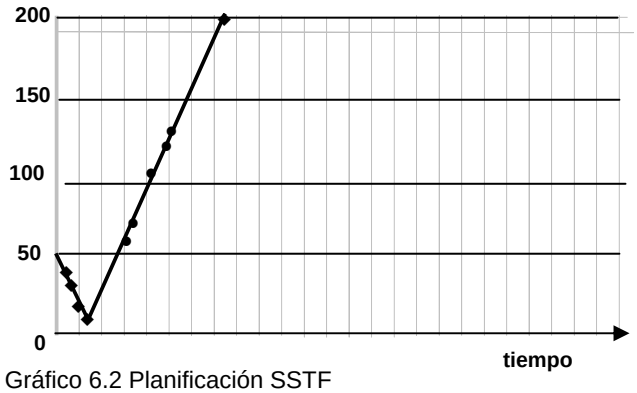


Gráfico 6.2 Planificación SSTF

El planificador SSTF es esencialmente una política similar a la que usa el planificador shortest-job-first (SJF), y, como éste planificador puede causar "inanición" de algún requerimiento, lo mismo ocurre con la planificación SSTF.

Supongamos que tenemos dos nuevos requerimientos en la cola, 15 y 185. Si llega un requerimiento cercano al 15 mientras estamos sirviendo a este requerimiento, este sería el próximo servido, haciendo esperar al requerimiento 185. Mientras este requerimiento comienza a ser servido, otro requerimiento cercano al 15 puede llegar. En teoría, una corriente continua de requerimientos cercanos uno de otro puede llegar, causando una espera indefinida para el requerimiento mas alejados como el 185. Esta escena es estadísticamente improbable, pero es posible.

Finalmente, el algoritmo SSTF, aunque es una importante mejora sobre el algoritmo FCFS, no es óptimo porque produce postergación indefinida (starvation).

Atiende primero los requerimientos cercanos al cabezal, puede causar Starvation de algunos pedidos.

Planificador SCAN

Cola de Pedidos: 100, 200, 45, 120, 10, 130, 60, 70, 30 y 20
Cola de Atención de Pedidos: 45, 30, 20, 10, 0, 60, 70, 100, 120, 130 y 200
Posición de arranque de las cabezas: 50 en dirección a la pista 0.
Cantidad de pistas recorridas por las cabezas: 250

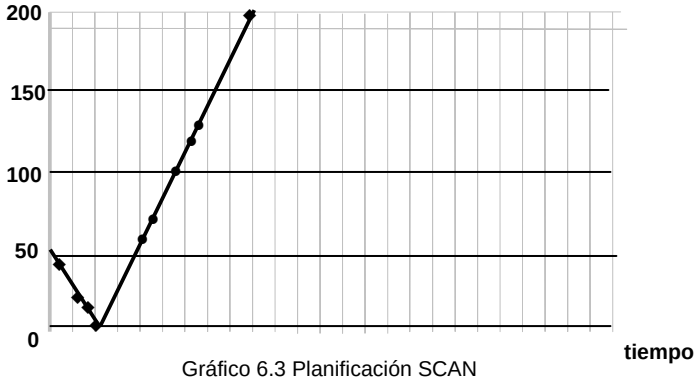


Gráfico 6.3 Planificación SCAN

El reconocimiento de la naturaleza dinámica de la cola de requerimientos conduce al algoritmo de búsqueda. La cabeza lecto-grabadora comienza en un extremo del disco, y se mueve hacia el otro extremo, sirviendo los requerimientos como estos van llegando pidiendo cada pista, hasta alcanzar el otro extremo

del disco. En el otro extremo, la dirección del movimiento de la cabeza se invierte en dirección contraria y continua sirviendo. La cabeza continuamente se mueve sobre el disco de extremo a extremo.

Usemos otra vez nuestro ejemplo: antes de aplicar el SCAN para planificar:

100, 200, 45, 120, 10, 130, 60, 70, 30 y 20

Necesitamos conocer la dirección del movimiento de la cabeza, además de la última posición que sirvió la cabeza. Si la cabeza se está moviendo hacia la pista 0, la cabeza atendería todos los pedidos menores en ese sentido, por ejemplo las pistas 45, 30, 20, 10 y el 0, luego el movimiento de la cabeza se invertiría y se mueve hacia el otro extremo del disco, sirviendo al requerimiento de la 60, 70, 100, 120, 130 y 200, llegando hasta el tope. Si llega un requerimiento en la cola justo en frente de la cabeza, será servido casi inmediatamente, sin embargo si un requerimiento llega justo detrás de la cabeza tendría que esperar hasta que la cabeza llegue al otro extremo del disco, invierta su dirección, y vuelva, antes de ser servido.

Asumiendo una distribución uniforme de los requerimientos por tracks, consideremos la densidad de los requerimientos cuando la cabeza alcanza un extremo e invierte su dirección. En este punto, hay relativamente pocos requerimientos inmediatamente detrás de la cabeza, puesto que los tracks fueron servidos recientemente. La mayor densidad de requerimientos está en el otro extremo del disco. Estos requerimientos también esperan bastante pero no adolece de la postergación indefinida que presentaba el algoritmo anterior.

El cabezal comienza en un extremo del disco y se mueve hacia el otro extremo. En el camino va transfiriendo los pedidos. Mucho tiempo de espera para los pedidos detrás del cabezal.

Planificador C-SCAN

Cola de Pedidos: 100, 200, 45, 120, 10, 130, 60, 70, 30 y 20
Cola de Atención de Pedidos: 60, 70, 100, 120, 130, 200, 0, 10, 20, 30 y 45
Posición de arranque de las cabezas: 50
Cantidad de pistas recorridas por las cabezas: 195

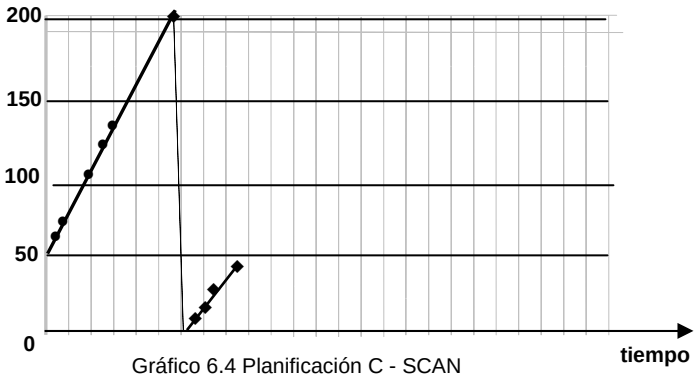


Gráfico 6.4 Planificación C - SCAN

Una variante del planificador SCAN que esta diseñada para proveer un tiempo de espera más uniforme es el planificador C-SCAN (circular SCAN). Como hace el planificador SCAN, el C-SCAN mueve la cabeza desde un extremo del disco hasta el otro, sirviendo los requerimientos solamente en ese sentido del movimiento. Cuando alcanza el otro extremo, retorna en forma brusca inmediatamente al comienzo del disco, sin servir ningún requerimiento en el viaje de vuelta que dura una fracción despreciable de tiempo. El algoritmo planificador C-SCAN trata esencialmente al disco como si fuera circular, con el último track adyacente al primero.

Planificador LOOK-UP o Algoritmo del Ascensor

Es semejante al SCAN, pero sólo llegan hasta el último requerimiento y no hasta el final del disco o sea su última pista.

Las cabezas se mueven en una dirección hasta que terminan los pedidos en esa dirección, entonces cambian la dirección y realizan los otros pedidos. Solo necesita que el software mantenga un bit indicando la dirección y que la cola esté ordenada en forma ascendente (para el movimiento de pista cero a máxima) y en forma descendente (para el movimiento inverso).

Nótese que, como los describimos, tanto el planificador SCAN como el C-SCAN siempre mueven la cabeza desde un extremo del disco hasta el otro, abarcando la totalidad de las pistas en su recorrido. En la práctica, ninguno de los dos algoritmos se implementa de este modo. Más comúnmente, la cabeza es sólo movida tan lejos como el último requerimiento en cada dirección. Tan pronto como no hay requerimientos en la actual dirección, el movimiento de la cabeza se invierte. Estas versiones del los planificadores SCAN y C-SCAN se llaman planificador LOOK-UP (mira por un requerimiento antes de moverse en esa dirección al igual que un ascensor) y planificador C-LOOK - UP.

Cola de Pedidos: 100, 200, 45, 120, 10, 130, 60, 70, 30 y 20
Cola de Atención de Pedidos: 45, 30, 20, 10, 60, 70, 100, 120, 130 y 200
Posición de arranque de las cabezas: 50 en dirección a la pista 0.
Cantidad de pistas recorridas por las cabezas: 230

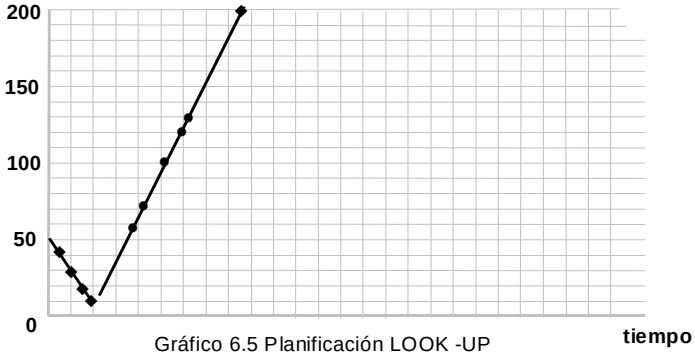


Gráfico 6.5 Planificación LOOK -UP

Planificador C-LOOK-UP o Algoritmo del Ascensor Modificado

Siempre atiende los pedidos en un solo sentido, desde el más bajo al número de cilindro del pedido mas alto.

Algunos controladores de discos proveen una manera de que el software inspeccione el número de sector que esta por debajo de la cabeza. Con estos controladores se puede hacer otra mejora. Si hay dos o mas pedidos para el mismo cilindro, el driver puede pedir la información que esta en el próximo sector que pasará por la cabeza sin mover el brazo.

Cuando hay varios pedidos (PES) para distintos discos, y uno de ellos esta leyendo o escribiendo los demás no pueden hacerlo. Cada driver tiene una lista de pedidos particular. Aunque no puede leer o escribir dos dispositivos simultáneamente, si puede adelantar la busque moviendo el brazo, preparándose para la operación (Esta afirmación no es válida para Controladores SCSI).

Como la tecnología avanza, los tiempos de búsqueda son cada vez mas cortos, pero el tiempo rotacional es siempre el mismo o a lo sumo mejora un pequeño porcentaje.

Otra solución es tener múltiples discos trabajando en paralelo. Una configuración interesante es tener, por ejemplo 38 drivers ejecutando en paralelo. Con una lectura se leen 38 bits al mismo tiempo, que forman una palabra de 32 bit con 6 bits de chequeo. Este diseño se llama RAID (Redundant Array of Inexpensive Disks).

Otras políticas pueden ser utilizando distintas administraciones de colas. Por ejemplo:

Cola de Pedidos: 100, 200, 45, 120, 10, 130, 60, 70, 30 y 20
Cola de Atención de Pedidos: 60, 70, 100, 120, 130, 200, 10, 20, 30 y 45
Posición de arranque de las cabezas: 50
Cantidad de pistas recorridas por las cabezas: 185

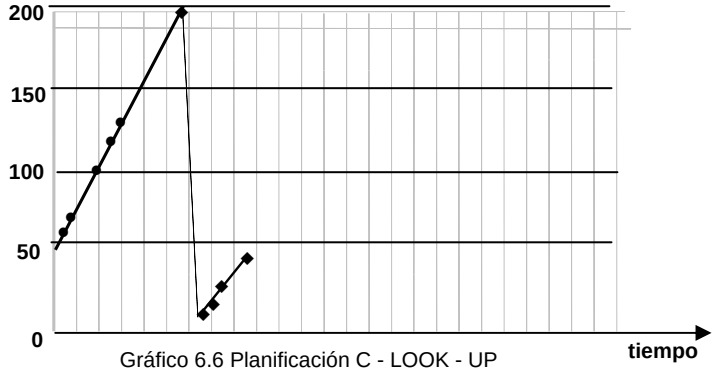


Gráfico 6.6 Planificación C - LOOK - UP

- **SCAN DE N PASOS**
  - divide la cola de peticiones del disco en subcolas de longitud N
  - las subcolas se procesan una a la vez mediante un SCAN
  - mientras se procesa una cola, se añadirán nuevas peticiones a las otras
- **FSCAN**

- dos colas
- todas las peticiones están en una de las colas y la otra permanece vacía

Planificador por Prioridad

Con un sistema de prioridades (PRI), el control de la planificación queda aislado del control del software gestor del disco. No se persigue la optimización del uso del disco, sino cumplir con otros objetivos del sistema operativo. Los trabajos por lotes que sean cortos y los trabajos interactivos reciben frecuentemente una prioridad más alta que trabajos mayores que realizan largas operaciones. Esta política podría conducir a contramedidas por parte de los usuarios, que pueden dividir sus trabajos en trozos más pequeños para explotar el sistema. Este tipo de política tiende a ser poco favorable para sistemas de bases de datos.

Planificador Ultimo en Entrar, Primero en Salir

Sorprendentemente, la política de tomar siempre la petición más reciente tiene alguna virtud. En los sistemas de proceso de transacciones, conceder el dispositivo al último usuario acarrea pocos o nulos movimientos del brazo al recorrer un archivo secuencial. El provecho de esta cercanía mejora la productividad y reduce la longitud de las colas. A medida que un trabajo utiliza de forma activa el sistema de archivos, va procesándose tan rápido como es posible. Sin embargo, si el disco está ocupado con una carga de trabajo larga, existe la posibilidad inconfundible de inanición.

La política FIFO, la de prioridades y el esquema LIFO (último en entrar, primero en salir) se basan únicamente en las propiedades de la cola o del proceso demandante. Si la posición de la pista actual es conocida por el planificador, puede emplearse una planificación en función del elemento demandado.

Nombre	descripción	Comentarios
Selección en función del Proceso demandante:		
FIFO	Primero en entrar, primero en salir	El más justo de todos
PRI	Prioridad del proceso	El control se lleva aparte de la gestión de la cola del disco
LIFO	Ultimo en entrar, primero en salir	Maximiza la utilización de recursos y aprovecha la cercanía
Selección en función del elemento solicitado por el proceso:		
SSTF	Primero el más corto	Gran aprovechamiento y colas pequeñas
SCAN	Recorrer el disco de un lado a otro	Mejor distribución del servicio
C-SCAN	Recorrer el disco en un sólo sentido	Menor variabilidad en el servicio
SCAN de *N* pasos	SCAN de *N* registros a la vez	Garantía de servicio
FSCAN	SCAN de *N* pasos, con *N* longitud de la cola al comienzo del ciclo del SCAN	Sensible a la carga

TABLA 6.3 Algoritmos de Planificación de Discos

Las políticas de **LIFO, FIFO** y **PRI**oridad se basan únicamente en las propiedades de la cola o del proceso demandante. Mientras que si la posición de la pista actual es conocida por el planificador, puede emplearse una planificación en función del elemento demandado utilizando SSTF, Scan, Lookup o sus variantes.

Selección de un Algoritmo Planificador de Disco

Hemos visto los algoritmos planificadores de disco, ¿cómo elegir un algoritmo en particular?

El algoritmo SSTF es común y tiene un interés natural. Los algoritmos planificadores SCAN y C-SCAN son más apropiados para sistemas que pongan una gran carga sobre el disco. Es posible definir un algoritmo óptimo, pero la computación necesaria para un planificador óptimo puede no justificar el ahorro sobre los planificadores SSTF o SCAN. Con cualquier algoritmo planificador, sin embargo, el rendimiento depende mucho del número y tiempo de requerimientos. En particular, si rara vez la cola tiene más de un requerimiento pendiente, entonces todos los algoritmos planificadores son efectivamente equivalentes. En este caso, el planificador FCFS es también un algoritmo razonable.

Nótese también que los requerimientos por servicios de disco pueden estar muy influenciados por el método de asignación de archivos. Un programa leyendo un archivo alojado en forma continua generará varios requerimientos que estarán todos juntos en el disco, resultando un reducido movimiento de la cabeza. Un archivo encadenado o indexado, al contrario, puede incluir bloques que están muy esparcidos sobre el disco, resultando una mejor utilización del espacio de disco y un gran movimiento de la cabeza.

La localización de los directorios y bloques de índices es importante también. Puesto que todos los archivos se deben abrir para usarlos, y la apertura de un archivo requiere buscar en la estructura de directorios, a los directorios se accede frecuentemente. Ubicando los directorios a mitad de camino entre

los bordes interiores y exteriores del disco, mejor dicho que cualquiera de los dos extremos, puede significar una reducción en el movimiento de la cabeza del disco. Por ejemplo, si una entrada al directorio está en el primer sector y un archivo de datos está en el último sector, entonces la cabeza del disco necesita moverse todo el ancho del disco. Si el registro del directorio está más hacia el medio, la cabeza tendrá que moverse a lo sumo la mitad del ancho.

Los algoritmos planificadores de disco, como todos los otros, pueden ser escritos como un módulo separado del sistema operativo, permitiendo ser removidos y reemplazados con un algoritmo diferente si fuera necesario.

Tampoco los planificadores FCFS o SSTF son una opción inicial razonable. En realidad los fabricantes de controladores de disco, han ayudado a los diseñadores de Sistemas Operativos incluyendo algoritmos planificadores del movimiento (PES) de la cabeza en el hardware mismo. El Sistema Operativo envía requerimientos al controlador en un orden FCFS, y el controlador los encola y los ejecuta en el modo más óptimo. Desafortunadamente, el Sistema Operativo puede ordenar los requerimientos en un modo que supone óptimo, y el controlador puede reacomodarlos en un orden distinto dando como resultando un decremento, en lugar de un incremento, del rendimiento.

En general se tiene en cuenta para comparar los algoritmos:

depende de
- Cantidad de pedidos
- Como están almacenados los archivos
- Posición de los directorios y sus archivos

* SSTF es bastante común y razonable pero produce starvation.
* SCAN & C-SCAN para sistemas con mucho uso de disco.
* Importante que el scheduler de disco sea un módulo independiente que pueda ser reemplazado.
* Si los cabezales son fijos, se puede hacer **sector queueing** (Cola por pista y sector)
  - ⇒ ordenar la cola de pedidos por número de sector para cada pista
  - ⇒ se hace una cola de pedidos por cada sector ⇒ cuando pasa el sector,

y esté bajo el cabezal, se atienden los pedidos para ese sector.

También, se puede usar con discos de cabezas móviles si hay más de un pedido para 1 pista particular.

Mejoras en velocidad y confiabilidad:

El Caché de disco puede estar implementado en Memoria Central, en la controladora del disco o en la unidad del disco:

* Caché de memoria: Independiente del disco
  Barato
  Pero usa Memoria Central y corre como otro proceso
* Caché de disco: Rapidísimo
  No usa Memoria Central
  No usa tiempo del microprocesador
  Depende del disco
  Usa muchos discos

Disk interleaving: se trata un grupo de discos como si fuera una única unidad de almacenamiento cada block se divide en los n discos.

Raid: Redundant Array of Inexpensive Disks

Mejora performance / precio y mejora confiabilidad porque provee duplicación de datos.

Mirroring o shadowing: Se mantiene una copia de cada disco.
Caro.

* Block Interleaved Parity:
  Los datos se escriben como en un disco normal, pero se escribe un block extra de paridad. Este bloque es la paridad de todos los bloques correspondientes de cada disco.
  Si se rompe un disco, se lo recompone usando paridad.
  Pero el block de paridad consume tiempo leerlo, grabarlo y calcularlo.

Caché de disco

Es un buffer en memoria central para sectores de disco, contiene una copia de algunos sectores del disco, generalmente muy accedido. Para administrar el espacio se utilizan las siguientes políticas:

Usado hace más tiempo (LRU)

- El bloque que ha permanecido sin referencias en la caché por más tiempo es reemplazado
- La caché consiste en una pila de bloques
- El bloque más recientemente referenciado está en la cima de la pila
- Cuando un bloque es referenciado o llevado a la caché, se pone en la cima de la pila

- Cuando se trae un bloque de la memoria secundaria se elimina el bloque que está en el fondo de la pila
- No es necesario estar moviendo estos bloques en la memoria
- Se tiene una pila asociada con la memoria caché con apuntadores a los bloques

La menos usada (LFU)

- El bloque que ha sufrido un menor número de referencias
- Se asocia un contador a cada bloque
- Con cada referencia al bloque, se incrementa el contador en 1
- Algunos bloques pueden ser referenciados muchas veces en un periodo corto de tiempo y después no necesitarse.

Reemplazo en función de la frecuencia

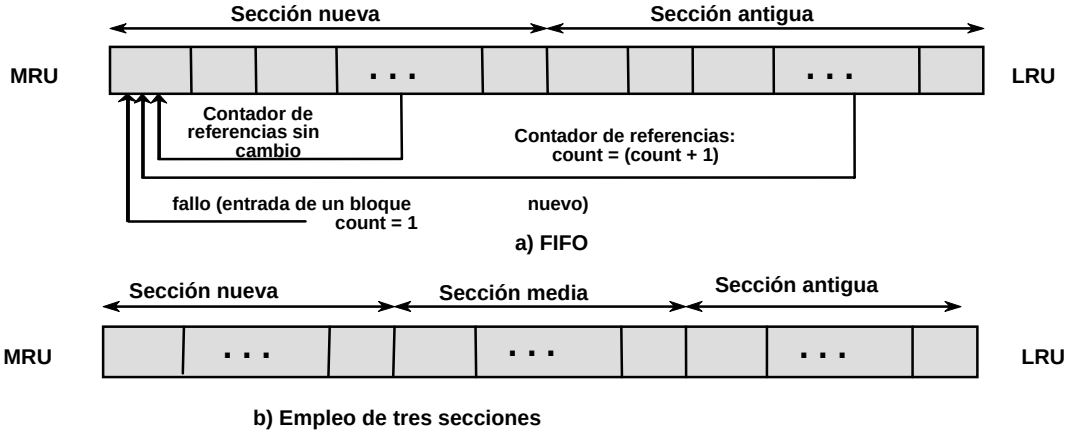


Fig. 6.14 a) política Fifo y b) empleo de 3 secciones de caché

Observación: MRU son las siglas de Most Recently Used (El mas reciente usado) y LRU significa Least Recently Used o sea el menos reciente usado.

Caché de una Pista por vez

Algunas unidades de discos tienen un caché de una pista, desconocido por el software independiente del dispositivo. Si se necesita un sector que está en el caché, no se necesita ninguna transferencia del disco. Una desventaja del caching de una pista por vez (aparte de la complejidad del software y el espacio del buffer que se necesita), es que la transferencia del caché al programa que lo pidió la tiene que hacer el procesador usando un LOOP programado, en lugar de dejar que la DMA del Hardware lo haga.

Algunos controladores llevan este sistema un paso mas adelante, y hacen el caching dentro de su propia memoria interna, así la transferencia entre el controlador y la memoria se puede hacer a través del DMA.

Las fallas en los discos causan la pérdida de información, y además la pérdida de tiempo y dinero, entonces se debe cambiar un disco por otro.

La información se resguarda por medio de backups, los cuales se deben hacer diariamente o en el final de la semana.

Han sido propuestas varias mejoras, en cuanto a las técnicas del uso del disco. Esos métodos involucran el uso de varios discos, trabajando cooperativamente. Para asegurar velocidad, se ha desarrollado una técnica llamada **DIVIDIR EN PEDAZOS AL DISCO (DISK STRIPING**). Un grupo de discos es tratado como una única unidad de almacenamiento, con cada bloque dividido en varios sub-bloques.

Cada sub-bloque se almacena en un disco separado. El tiempo que se necesita para transferir un bloque hacia la memoria disminuye drásticamente, puesto que los discos transfieren sus sub-bloques en paralelo. Esta disminución es notable si además los discos están sincronizados: es decir que no existe espera, entre cada acceso de un sub-bloque, en cada Unidad. Esta ventaja se ve magnificada, si además se utilizan discos pequeños y no costosos, en lugar de discos caros y grandes.

Esta idea fue la base para el desarrollo de arreglos redundantes de discos no costosos (RAID[3]), que mejoró la performance y garantizó la duplicación de datos para mejorar la seguridad.

[3] RAID inicialmente significaba Redundant Array of Inexpensive Disk, pero hoy se le asigna a Discos independientes en ves de baratos. El término RAID hizo su debut oficial en 1989 en forma de un paper publicado por **David Paterson, Garth Gibson y Randy Katz**, todos ellos de la Universidad de California. El paper se titulaba "A case for Redundant Array of Inexpensive Disks".

Arreglos de discos (RAID)

Se puede decir que los discos son la componente menos confiable de un computador, la más complicada de sustituir, y la que frena el mejoramiento de la velocidad de procesamiento con los avances tecnológicos. En efecto, la velocidad de los procesadores se duplica más o menos cada 2 años, y la capacidad de los chips de memoria crece a un ritmo parecido. No obstante, el ancho de banda (velocidad de transferencia) del I/O ha variado muy poco.

A este ritmo, en 7 años más los procesadores van a ser 10 veces más rápidos, pero en general las aplicaciones correrán menos de 5 veces más rápido, por las limitaciones de I/O.

Una solución posible para este problema es que en lugar de un solo disco grande, usar muchos discos chicos y baratos, en paralelo, para mejorar el ancho de banda. Para garantizar paralelismo, se hace *disk striping* o división en franjas. Cada bloque lógico se compone de varios sectores físicos, cada uno en un disco distinto. Así, cada acceso a un bloque lógico se divide en accesos simultáneos a los discos. Existe un gran problema que es la confiabilidad en fallas del disco. Para resolverlo se usa redundancia, en una configuración conocida como RAID (Redundant Array of Inexpensive Disks), y que se puede implementar en varios niveles.

RAID 1.

Se usan **discos espejos**, o sea, la información de cada disco se mantiene siempre duplicada en otro idéntico. O sea, MTTF aumenta notoriamente, pero duplicando el costo.

RAID 2.

Se reduce la redundancia usando técnicas de detección y corrección de errores (códigos de Hamming). Por ejemplo, si un bloque se reparte entre 10 discos y suponemos que no va a haber más de una falla simultáneamente, entonces no necesitamos duplicar el bloque entero para reconstituirlo en caso de falla, puesto que ante una falla sólo se perderá un 10% de la información. El problema es que si no sabemos qué 10% se perdió, de todas maneras se necesita bastante redundancia oscilando mas o menos entre el 20 y 40%.

RAID 3.

El punto es que, gracias a que cada controlador usa sumas de chequeo (y suponiendo que además podemos saber cuándo un controlador falla) sí podemos saber qué trozo de la información está errónea. Y sabiendo eso, basta con usar sólo un disco adicional para guardar información de paridad con la cual es posible reconstituir la información original.

RAID 4

Tanto el nivel 4 como el 5 hacen uso de técnicas de acceso independientes. Aquí cada miembro del disco opera independientemente donde los pedidos de E/S separados pueden ser satisfechos en paralelo. Este RAID mas específicamente envuelve una penalidad de escritura cuando un pedido de escritura de un E/S es ejecutado. Cada vez que ocurre una escritura el software del administrador de array debe actualizar no solo el dato del usuario, si no que también la paridad de los bits.

RAID 5

Esta organizado en un a forma similar al RAID 4. La diferencia radica en que este RAID distribuye las listas de paridades a través del disco. Una típica asignación es un esquema de round-robin. Esta distribución a través de todas las unidades evita el amontonamiento de potencial que encontramos en el RAID anterior.

Con el Uso de múltiples discos, hay una amplia variedad de formas en las cuales los datos pueden ser organizados y en las cuales la redundancia puede ser agregada para mejorar la confiabilidad. Esto podría hacer dificultoso el desarrollo de planes de bases de datos que son usados en un número de plataformas y sistemas operativos. Afortunadamente, la industria acordó un plan estandarizado para el diseño de bases de datos de discos múltiples, conocido como RAID. El plan RAID consiste en 7 niveles (del 0 al 6). Estos niveles no implican una relación jerárquica pero designan diferentes diseños de arquitectura que comparten tres características comunes:

- RAID es un conjunto de discos físicos vistos por el sistema operativo como una unidad de disco lógica
- Los datos son distribuidos a través de las unidades físicas de un arreglo
- La capacidad redundante de disco es usada para almacenar información de la paridad, lo cual garantiza la recuperación de los datos en caso de falla del disco.

La versión más sencilla de RAID, consiste en mantener una copia duplicada de cada disco. Esta solución es costosa y se conoce como RAID - 1.

En la versión más compleja, la información se graba en cada disco, en el array en una unidad de bloque, como en las configuraciones normales. Además se graba un bloque extra de paridad. Este bloque de paridad, representa la paridad de todos los bloques equivalentes en cada disco, en el array. Por

ejemplo, si hay 8 discos en el array, entonces el sector 0 del disco 1 hasta el 7, tiene su paridad calculada y almacenada en el disco 8.
Como la relación entre seguridad y velocidad es importante, muchos vendedores investigan y producen hardware y software RAID.

Nota: Además de estos niveles de RAID algunos hacen mención al RAID 0, sin embargo este no pertenece a la familia de los RAIDs ya que justamente no usa redundancia.

A continuación se describen en resumen los niveles en la tabla 6.4 y sus principales características:

Categoría	Nivel	Descripción	Tasa de Pedido de E/S (Lectura/Escritura)	Tasa de Transferencia de Datos	Aplicación Típica
Separación	0	No redundante	Separaciones grandes: Excelente	Separaciones pequeñas: Excelente	Aplicaciones requieren alto rendimiento para datos no críticos
Mirroring	1	Mirrored	Bueno/Justo	Justo	Unidades de Sistema; archivos críticos
Acceso Paralelo	2	Redundante por código de Hamming	Pobre	Excelente	?
	3	Paridad de Bit Intercalado	Pobre	Excelente	Pedidos grandes de E/S como aplicaciones CAD
Acceso Independiente	4	Paridad de Bloque intercalado	Excelente/Justo	Justo/Pobre	?
	5	Paridad distribuida de Bloque intercalado	Excelente/Justo	Justo/Pobre	Alta tasa de Pedido, lectura intensiva, búsqueda de datos
	6	Paridad doblemente distribuida de bloque intercalado	Excelente/Pobre	Justo/Pobre	Aplicaciones requieren extremadamente alta disponibilidad

Tabla 6.4. Características de los arreglos de discos.

6.4.5. Almacenamiento intermedio de E/S (Buffering)

Un **buffer** es un espacio de memoria reservado para almacenamiento temporal de datos, para acomodar disparidades entre las velocidades de un dispositivo productor de datos y un dispositivo consumidor, o diferencias en tamaños de bloques.

Con las disparidades entre las velocidades existen dos problemas: el primero, es que el programa esta colgado esperando que se complete la operación de E/S. El segundo problema es que esto interfiere con las decisiones del swapping a través del sistema operativo.

Para evitar este tipo de ineficiencias y otras de otro tipo a veces es conveniente ejecutar
Las transferencias de entrada a medida que se fueron haciendo los pedidos ejecutar transferencias de salida una vez que el pedido ya este hecho. Esta técnica se conoce como **buffering.**

Entonces, un **buffer** es un área de memoria en la que se almacenan datos temporalmente mientras se transfieren entre dos dispositivos o entre un dispositivo y una aplicación.

Cuando un proceso realiza un pedido de E/S, es bloqueado hasta que la operación se termine, pero no puede ser intercambiado ya que las posiciones desde o hacia las cuales debe hacerse la transferencia deben continuar en memoria central para ser utilizadas. Para evitar esto a veces es conveniente llevar a cabo las transferencias de entrada por adelantado a las peticiones y realizar las transferencias de salida un tiempo después de la petición. Esta técnica se conoce como almacenamiento intermedio (I/O buffering).

Las razones para el almacenamiento intermedio son:

- Los procesos deben esperar que una E/S termina
- Ciertas páginas deben permanecer en memoria central durante una E/S.

Orientado a bloque

- Almacenan la información en bloques de tamaño fijo
- Las transferencias son de un bloque a la vez
- Se usa en discos y cintas

Orientado a flujo

- Transfieren los datos como flujos de bytes

- Se usa en terminales, impresoras, puertos de comunicación, ratones, y muchos otros dispositivos que no son de almacenamiento secundario.

Sin Almacenamiento Intermedio:

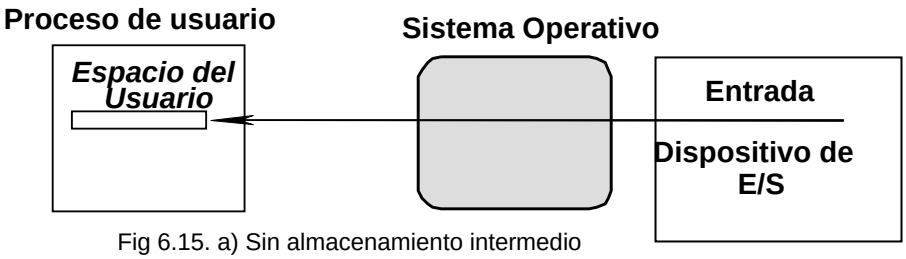


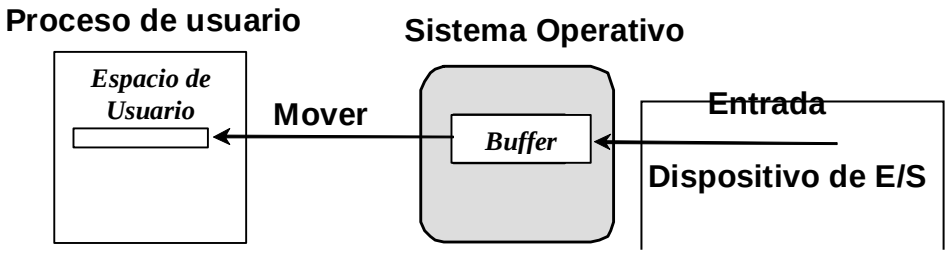
Fig 6.15. a) Sin almacenamiento intermedio

Buffer Sencillo o único

Esta técnica se implementa mediante un único buffer en el cual se va guardando la información a medida que es transferida. Cuando una determinada porción de información fue copiada, es movida luego al espacio de memoria del proceso y se pide la transferencia de otra porción. Cuando se realiza una salida, la técnica se efectúa justo a la inversa.

Cuando un proceso realiza una petición de E/S, el S.O. le asigna a la operación un buffer en la parte del sistema de la memoria central. Para los dispositivos de bloques, las transferencias de entrada se realizan al buffer del sistema, cuando se ha completado la transferencia el proceso mueve el bloque al espacio del usuario y pide otro bloque. Esta técnica es **llamada lectura por adelantado** y se realiza esperando que el bloque se necesite más adelante. De esta forma el S.O. será capaz de expulsar al proceso porque la operación de entrada tiene lugar dentro de la memoria del sistema. Para la E/S con dispositivos de flujo el esquema de buffer sencillo puede aplicarse por líneas o por bytes.

- El S.O. asigna un buffer en memoria central para una solicitud de E/S
- Orientado a bloques
  - Las transferencias de entrada se realizan en el buffer
  - Se mueve el bloque al espacio del usuario cuando se necesita
  - Se mueve otro bloque al buffer: lectura adelantada



Mover: Al completarse la transferencia, el proceso mueve el bloque y pide otro inmediatamente.

Técnica para dispositivos de bloques: lectura por adelantado o entrada anticipada

Fig. 6.15 b)Buffer sencillo

Orientado a Bloque

- El proceso de usuario puede procesar un bloque de datos mientras se esté leyendo el siguiente bloque de datos
- El intercambio puede ocurrir ya que la entrada está tomando lugar en la memoria del sistema, no en la memoria del usuario
- El S.O. mantiene constancia de las asignaciones de buffers del sistema a los procesos de usuario
- La salida se realiza por el proceso de usuario escribiendo un bloque el buffer y después hacia fuera

Orientado a Flujo

- Usa una línea a la vez
- La entrada del usuario de una terminal es una línea a la vez con **cr** (carrier return) indicando el fin de la línea
- La salida a la terminal es una línea a la vez

Buffer Doble

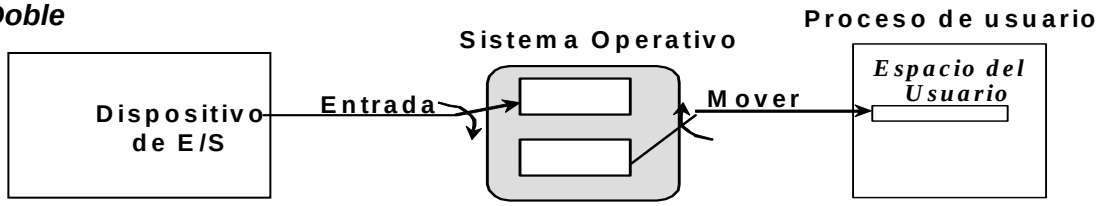


Fig. 6.15. c) Doble buffer

Como su nombre lo indica en esta técnica se utilizan dos buffers para realizar la función de E/S. Los procesos transfieren datos desde/hacia un buffer mientras el sistema operativo llena o vacía (según corresponda) el otro buffer. De esta forma el proceso no tendrá que esperar por la E/S, ganando velocidad de procesamiento.

Entonces se asignan a un proceso dos buffers del sistema a la operación de E/S. De esta forma el proceso puede transferir datos hacia o desde un buffer mientras que el S.O. vacía o llena al otro. Esta técnica mejora las transferencias de bloques y de flujos por línea, pero en la operación de flujos por bytes no ofrece ventajas con respecto a un buffer sencillo de doble tamaño.

Buffer Circular

Esta técnica se utiliza si el proceso realiza frecuentes ráfagas de E/S, lo que se hace es permitir que el proceso utilice más de dos buffers. Cada buffer individual se considera como una unidad del buffer circular, y son tratados según el modelo de productor/consumidor. Con esta técnica se puede maximizar el desempeño individual del proceso en cuestión.

- Se usan más de dos buffers, cada buffer individual es una unidad en un buffer circular.
- Se basa en el modelo productor consumidor con buffer limitado.

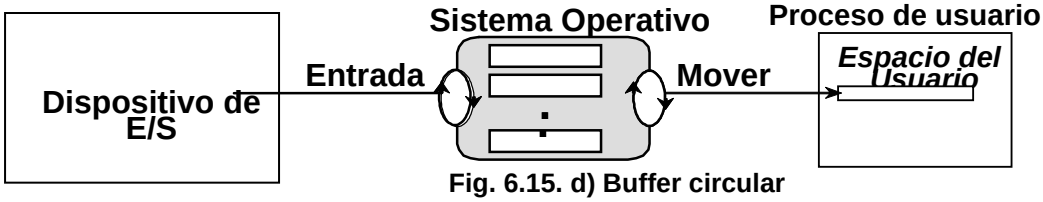


Fig. 6.15. d) Buffer circular

El esquema del buffer doble debería solucionar el flujo de datos entre un dispositivo de E/S y un proceso. Si preocupa el rendimiento de un proceso determinado, sería deseable que las operaciones de E/S fueran capaces de ir al ritmo del proceso. El buffer doble puede ser inapropiado si el proceso lleva a cabo rápidas ráfagas de E/S. En este caso, el problema puede mitigarse usando más de dos buffers. Cuando se emplean más de dos, el conjunto de buffers se conoce con el nombre de *buffer circular* (figura 6.15d). Cada buffer individual constituye una unidad del buffer circular. Este se basa en el modelo del productor/consumidor con un buffer limitado.

La Utilidad del Almacenamiento Intermedio

El almacenamiento intermedio es una técnica que soluciona los problemas de "cuellos de botella" en la demanda de E/S. Sin embargo, no existe un tamaño de los buffers que asegure a un dispositivo de E/S ir al mismo ritmo que un proceso cuando la demanda media del proceso es mayor que la que el dispositivo puede admitir. Incluso si se dispone de varios buffers, al final todos se llenarán y el proceso tendrá que quedarse esperando tras operar con una determinada cantidad de datos. Sin embargo, en un entorno de multiprogramación, con la variedad de actividades de E/S y de procesamiento que hay que realizar, el almacenamiento intermedio es una herramienta que puede incrementar la eficiencia del sistema operativo y el rendimiento de los procesos individuales.

Spooling.

Un spool es un buffer que contiene la salida para un dispositivo de caracteres, tal como una impresora, en el cual no se pueden mezclar las salidas de varios procesos. Mediante spooling, los procesos tienen la ilusión de estar imprimiendo simultáneamente, pero en realidad el sistema operativo está almacenando la salida de cada proceso para imprimirla de una sola vez cuando el proceso termine.

6.4.6. Dispositivos Internos.

Discos RAM

Usa una porción de memoria pre-asignada para almacenar los bloques. Tiene la ventaja de que el acceso es instantáneo (a la velocidad de la memoria central). El disco simulado en memoria se divide en bloques del mismo tamaño que los del disco real.

Clocks (Timers)

Son esenciales en los sistemas de tiempo compartido. Evitan que los un proceso monopolice el uso de la CPU, entre otras cosas.

Hardware del Clock:

En las computadoras se usan comúnmente dos tipos de clocks. Los mas simples están conectados con una línea de 110 o 220 volts, y provocan una interrupción con cada ciclo de tensión, cada 50 o 60 Hz (Hertz o ciclos por segundo).

El otro tipo de clock se construye de tres componentes: un oscilador de cristal piezoeléctrico como el cuarzo, un contador y un registro de tenencia. El cristal de cuarzo genera una señal periódica muy exacta, normalmente dentro de un rango de 5 a 100 MHz, dependiendo del cristal. Cada vez que se recibe una señal el contador se decrementa en uno. Cuando el contador llega a cero, se produce una interrupción.

Los clocks programables tienen varios modos de operación:

En **UNO SOLO DISPARO,** cuando se inicia el clock, copia el valor del registro de tenencia, y se decrementa el contador con cada pulso. Cuando el contador llega a cero, se produce una interrupción y se para, hasta que lo inicien de nuevo.

MODO DE ONDA CUADRADO (square-wave), luego de que llega a cero, el valor del registro de tendencia se pone en el contador automáticamente, y este proceso se repite indefinidamente. Estas interrupciones periódicas se llaman CLOCK TICKS.

Para utilizar un clock del día (time-of-day clock), el software le pide al usuario el tiempo actual, que se traduce en un número de ticks, por ejemplo UNIX lo hace desde la cero hora del 01-01-1970. Algunas computadoras tienen registros especiales para guarda el tiempo actual, que son alimentados por una batería (battery backup).

Software del Clock

Funciones del driver del clock:

- Mantener el día
- Evitar que algunos procesos se ejecuten mas de lo asignado
- Contabilizar el uso de la CPU
- Manejar las llamadas de sistema de ALARMA hechas por los programas del usuario

Mantener el día Hay tres formas de hacerlo:

a) Usar un contador de 64 bits, contando los ticks.
b) Mantener el día en segundos. Para esto se necesita otro contador que acumule los ticks hasta un segundo, y luego allí hay que aumentar el contador.
c) Contar en tics, pero desde el momento en que el sistema se bootea. Cuando el usuario ingresa el tiempo se calcula el tiempo desde el valor del time-of-day actual y se almacena en memoria del la manera mas conveniente. Luego cuando se necesita, se suma al contador el valor almacenado para obtener el actual.

Para evitar que algunos procesos se ejecuten mas de lo asignado se procede:

- Cuando un proceso comienza a ejecutarse se pone en el contador el valor del quantum asignado a ese proceso.

- Contabilizar el uso del procesador. Hay dos manera de hacerlo:

a) Iniciar un clock secundario cada vez que un proceso comienza a ejecutarse, cuando el proceso se detiene se salva el valor, para calcular cuanto tiempo se ejecutó.
b) Mas simple, pero menos exacto, es mantener un puntero en la entrada a la tabla de procesos, en un campo especial se incrementa cada vez que se produce un tick (si se producen demasiadas interrupciones, parecería que el proceso se ejecutó mucho, pero en realidad, la mayoría del tiempo lo usó el S.O. para manejar la interrupciones).

Algunas partes del sistema también necesitan timers, estos se llaman WATCHDOG TIMERS.

El mecanismo que se usa para manejar los watchdogs timers es el mismo que con las señales, pero en lugar de causar una interrupción, el driver del clock llama a un procedimiento que provee el servicio para el que lo llamó. El procedimiento es parte del código del que lo llamó.

6.5. TÉCNICAS DE E/S

Hay tres técnicas posibles para las operaciones de E/S. Con la ***E/S Programada*, también llamada de *Polling (Escrutinio)***, los datos son intercambiados entre el procesador y el módulo de E/S. El procesador ejecuta el programa que otorga el control directo de la operación de E/S, incluyendo el sensado del estado del dispositivo, enviar un comando READ o WRITE, y la transferencia de los datos. Cuando el procesador emite un comando al módulo de E/S, debe esperar hasta que la operación de E/S se complete. Si el procesador es más rápida que el módulo de E/S, se desperdicia tiempo de la CPU. Con **la *E/S dirigida por interrupciones*** (interrupt driven), el procesador emite un comando de E/S, continúa con la ejecución de otras instrucciones, y es interrumpida por el módulo de E/S cuando éste ha completado su trabajo. Tanto en la E/S programada o por interrupciones, el procesador es responsable de extraer los datos de la memoria central para el output o almacenar los datos en la memoria central para el input. La alternativa a esto se conoce como ***acceso directo a memoria* (DMA).** En este modo, el módulo de E/S y la memoria central intercambian datos directamente, sin involucrar a la CPU.

Transferencia de datos:	Sin Interrupciones	Con uso de Interrupciones
desde E/S a Memoria a través de la CPU	E/S Programada	E/S por Interrupciones
Directa entre Módulo E/S y Memoria	----------	Acceso Directo a Memoria (DMA)

Tabla 6.5. Técnicas de E/S

La Tabla 6.5 indica la relación entre éstas tres técnicas. Vemos primero la E/S programada, en segundo lugar la E/S por interrupciones y finalmente el Acceso Directo a Memoria (DMA).

6.5.1. E/S Programada

Cuando el procesador está ejecutando un programa y encuentra una instrucción relativa a E/S, ejecuta esa instrucción emitiendo un comando al módulo de E/S apropiado.

Con la E/S programada, el módulo de E/S ejecutará la acción solicitada y luego establecerá los bits apropiados en el registro de status de E/S (ver Figura 6.16). El módulo de E/S no ejecuta ningún tipo de acción para alertar a la CPU. En particular, no interrumpe a la CPU. Luego, es responsabilidad del procesador de chequear periódicamente el status del módulo de E/S hasta que encuentra que la operación solicitada ha concluido.

Para explicar la técnica de E/S programada, la vemos primero desde el punto de vista de los comandos emitidos por el procesador al módulo de E/S, y luego desde el punto de vista de las instrucciones de E/S ejecutadas por la CPU.

Comandos de E/S Programada

Para ejecutar una instrucción del tipo de E/S, el procesador emite una dirección, especificando el módulo en particular y el dispositivo externo, y un comando de E/S. Hay cuatro tipos de comandos que un módulo de E/S puede recibir cuando está direccionado por la CPU. Estos se clasifican como de **control, de verificación (test), read y write.**

Un comando de *control* es usado para activar un periférico y decirle qué hacer. Por ejemplo, una unidad de cinta (magnética) puede ser instruida a rebobinar o avanzar un registro. Estos comandos son diseñados para el tipo particular de dispositivo.

Un comando de *verificación* (test) es usado para verificar las condiciones de estados asociado con un módulo de E/S y sus periféricos. El procesador querrá saber que el periférico de interés está operativo y disponible para su uso. También querrá saber si la operación más reciente de E/S fue completada y si algún tipo de error ocurrió.

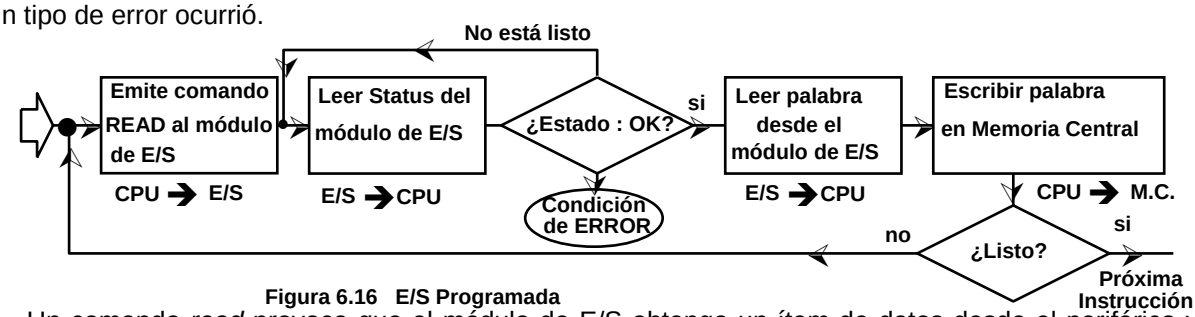


Figura 6.16 E/S Programada

Un comando *read* provoca que el módulo de E/S obtenga un ítem de datos desde el periférico y lo coloque en el buffer interno (representado como un registro de datos en la Figura 6.02 o 6.07). El procesador puede luego obtener el ítem de datos solicitando al módulo de E/S lo coloque en el bus de

datos. Inversamente, un comando *write* causa que el módulo de E/S tome un ítem de datos (un byte o una palabra) del bus de datos y subsecuentemente lo transmita al periférico.

La Figura 6.16 da un ejemplo del uso de la E/S programada para leer un bloque de datos desde un dispositivo periférico (por ejemplo, un registro de una cinta) en la memoria. Los datos son leídos en una palabra (por ejemplo, 16 o 32 bits) a la vez. Para cada palabra leída, el procesador debe permanecer en un ciclo de verificación de estados hasta que determina que la palabra está disponible en el registro del módulo de E/S. Este diagrama de flujo resalta la principal desventaja de esta técnica: es un proceso que consume tiempo que mantiene al procesador ocupado innecesariamente.

Instrucciones de E/S

Con la E/S programada, hay una estrecha correspondencia entre las instrucciones del tipo E/S que el procesador obtiene (fetch) de la memoria y los comandos de E/S que el procesador emite al módulo de E/S para ejecutar las instrucciones. Esto es, las instrucciones son fácilmente relacionadas con los comandos de E/S y, a menudo, se puede establecer una correspondencia uno-a-uno entre las instrucciones y los comandos de E/S.

Típicamente, habrá varios dispositivos de E/S conectados a través de módulos de E/S al sistema. A cada dispositivo se le asigna un único identificador o dirección. Cuando el procesador emite un comando de E/S, el comando contiene la dirección del dispositivo deseado. Luego, cada módulo de E/S debe interpretar las líneas (del bus) de direcciones para determinar si el comando es para el mismo.

Cuando la CPU, la memoria central, y el módulo de E/S comparten un bus común, dos modos de direccionamiento son posibles: *por* ***correspondencia en memoria*** (memory-mapped) y *aislado*. En el caso de correspondencia de E/S en memoria, hay un único espacio de direccionamiento para la memoria y los dispositivos de E/S. El procesador trata los registros de estados y de datos de los módulos de E/S como locaciones en memoria y utiliza las mismas instrucciones de máquina para acceder tanto a la memoria como a los dispositivos de E/S. De esta manera, por ejemplo, con 10 líneas de direcciones, una combinación total de 1024 locaciones y direcciones de E/S pueden ser soportadas, en cualquier combinación.

Con la E/S por correspondencia en memoria (memory-mapped), una única línea de read y una única línea de write son necesarias en el bus. Alternativamente, el bus puede estar equipado con read y write para memoria más líneas para input y output de comandos. Ahora, la línea de comando especifica si la dirección referencia a una locación en memoria o a un dispositivo. El rango completo de direcciones puede estar disponible para ambos. Otra vez, con 10 líneas de direcciones, el sistema puede ahora soportar 1024 locaciones en memoria y 1024 direcciones de E/S. Dado que el espacio de direcciones está aislado del espacio de direcciones de la memoria, es que a ésta técnica se la conoce como *E/S aislada*.

DIRECCIÓN	INSTRUCCIÓN	OPERANDO	COMENTARIO
200	Load AC	"1"	
	Save AC	517	**Inicia Lectura de Entrada de Teclado**
202	Load AC	517	**Obtener el Byte de Estado**
	Branch if Sign=0	202	**Iterar hasta que esté listo**
	Load AC	516	

a) E/S POR CORRESPONDENCIA EN MEMORIA

DIRECCIÓN	INSTRUCCIÓN	OPERANDO	COMENTARIO
200	Start I/O	5	**Inicia Lectura de Entrada de Teclado**
201	Test I/O	5	**Chequea por terminación**
	Branch Next Ready	201	**Iterar hasta completar**
	In	5	**Cargar Byte de Dato**

b) E/S AISLADA

Figura 6.17 E/S por correspondencia en Memoria y aislada

La Figura 6.16 contrasta estas dos técnicas de E/S programada. La Figura 6.17-a muestra cómo la interfase de un dispositivo de entrada sencillo como el teclado de una terminal puede aparecer al programador utilizando E/S por correspondencia en memoria (memory-mapped). Asuma direcciones de 10 bits, con una memoria de 512 bytes (posiciones 0-511) y hasta 512 direcciones de E/S (posiciones 512-1023). Dos direcciones son dedicadas al input del teclado para una terminal en particular. La dirección 516 refiere al registro de datos y la dirección 517 refiere al registro de estados, el cual también trabaja como registro de control para recibir los comandos de la CPU. EL programa mostrado leerá 1 byte de datos del teclado en un registro acumulador en la CPU. Observe que el procesador se queda en el bucle hasta que el dato (byte) este disponible.

Con la E/S aislada (Figura 6.17-b), los puertos de E/S son accesibles únicamente por medio de comandos especiales de E/S, los cuales activan las líneas de comando en el bus.

Para la mayoría de los tipos de CPU, hay un relativamente extenso conjunto de instrucciones para referenciar a la memoria. Si se utiliza E/S aislada, hay sólo unas pocas instrucciones de E/S. En consecuencia, una ventaja de la E/S por correspondencia en memoria (memory-mapped) es el largo repertorio de instrucciones que pueden ser utilizadas, permitiendo una programación más eficiente. Una desventaja es que un valioso espacio de direccionamiento es utilizado. Tanto la E/S por correspondencia en memoria como la aislada son de uso común.

6.5.2. E/S por Interrupciones

El problema con la E/S programada es que el procesador tiene que esperar un tiempo considerable para que el módulo de E/S concerniente esté listo para la recepción o transmisión de datos. La CPU, mientras está esperando, debe repetidamente interrogar por el estado del módulo de E/S. Como resultado, el nivel de performance de todo el sistema es severamente degradado.

En la Figura 6.26 se pueden observar con mayor detalle los pasos y controles para una operación de E/S por interrupciones.

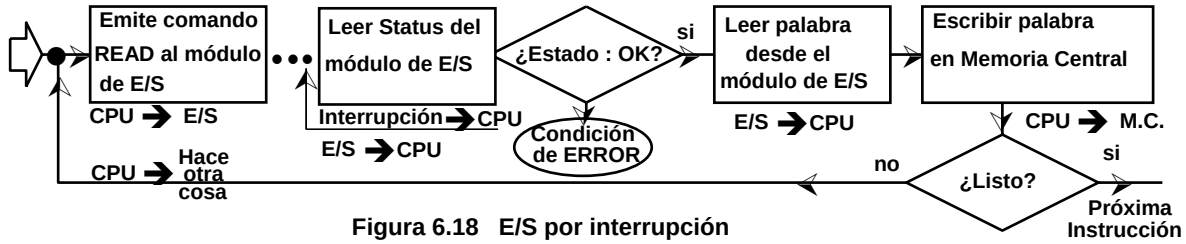


Figura 6.18 E/S por interrupción

Desde el punto de vista de la CPU, la acción para un input es como sigue. El procesador emite un comando READ. Luego se "apaga" y hace otra cosa (por ejemplo, el procesador podría estar trabajando en varios programas diferentes al mismo tiempo). Al final de cada ciclo de instrucción, el procesador verifica si hubo alguna interrupción (Figura 6.18). Esta verificación lo hace el hardware y es muy veloz.

Cuando la interrupción desde un módulo de E/S ocurre, el procesador salva el contexto de ejecución (el contador de programa y los registros de la CPU) del programa corriente y procesa la interrupción. En este caso el procesador lee la palabra de datos del módulo de E/S y la almacena en memoria. Luego recupera el contexto del programa que estaba ejecutando (u otro programa) y continúa su ejecución.

Una alternativa es que el procesador emita el comando de E/S a un módulo y luego ejecute otro trabajo útil. El módulo de E/S interrumpirá luego al procesador para solicitarle el servicio cuando está lista para el intercambio de datos con la CPU. El procesador ejecuta luego la transferencia de datos, como antes, y luego continúa con el procesamiento que había quedado pendiente.

Consideremos como trabaja esto, primero desde el punto de vista del módulo de E/S. En el caso de una operación de entrada, el módulo recibe un comando READ desde la CPU. El módulo de E/S procede luego a leer datos provenientes de un periférico asociado. Una vez que los datos están en los registros de datos del módulo, el módulo señaliza una interrupción al procesador sobre una línea de control.

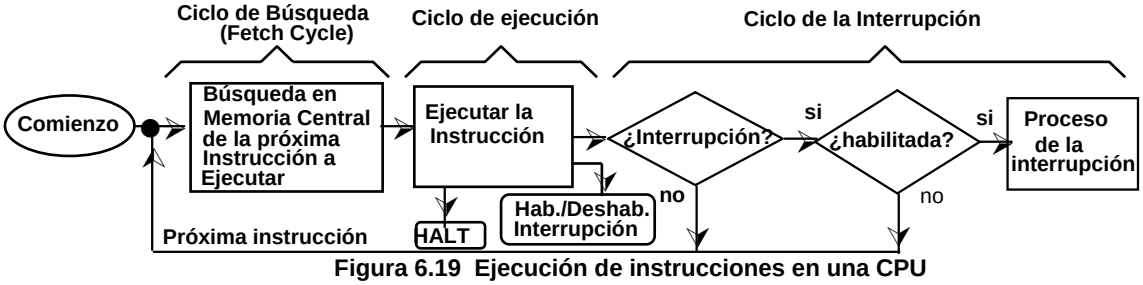


Figura 6.19 Ejecución de instrucciones en una CPU

El módulo espera hasta que sus datos sean solicitados por la CPU. Cuando la solicitud está hecha, el módulo coloca sus datos en el bus de datos y luego está listo para otra operación de E/S.

La Figura 6.18 muestra el uso de la E/S por interrupción para la lectura de un bloque de datos. Compare esto con la Figura 6.16.

La E/S por interrupción es más eficiente que la E/S programada porque elimina la espera innecesaria del procesador por la E/S. Sin embargo, la E/S por interrupción todavía consume un gran cantidad de tiempo de CPU, dado que cada palabra de datos que vaya de la memoria al módulo de E/S o al revés, debe pasar por la CPU. Ella es la responsable de la transferencia de los datos.

6.5.3. E/S por DMA (ACCESO DIRECTO A MEMORIA)

Desventajas de la E/S por Interrupción o Programada

La E/S por interrupciones, si bien es más eficiente que la E/S programada, todavía requiere de la intervención activa del procesador para transferir los datos entre la memoria y el módulo de E/S, y cualquier transferencia de datos debe recorrer un camino pasando por la CPU. Luego, ambas formas de E/S sufren de dos desventajas inherentes:

- La velocidad de transferencia de E/S está limitada por la velocidad a la cual el procesador puede verificar y servir a un dispositivo.
- El procesador está ocupada administrando la transferencia de E/S; un número de instrucciones debe ser ejecutadas para cada transferencia de E/S (por ejemplo, ver Figura 6.20)

Hay cierto grado de compromiso entre estas dos desventajas. Considere la transferencia de un bloque de datos. Usando E/S programada, el procesador es dedicado a la tarea de E/S y puede mover los datos a una relativamente alta velocidad, al costo de no hacer nada más. La E/S por interrupción libera al procesador hasta cierto punto a expensas de la velocidad en E/S. No importa cual, ambos métodos tienen un impacto adverso tanto en la actividad del procesador como en la velocidad de transferencia de E/S.

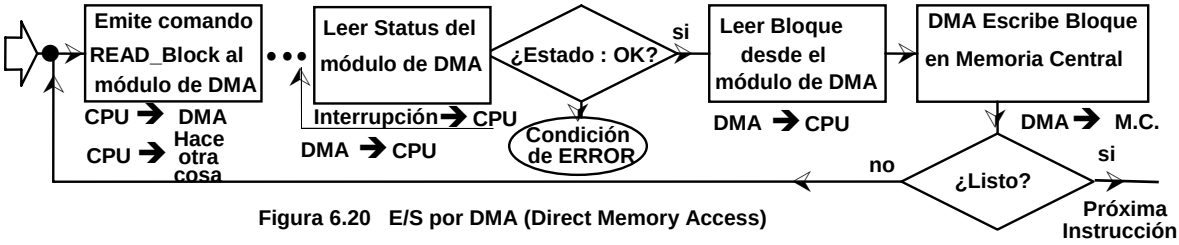


Figura 6.20 E/S por DMA (Direct Memory Access)

Cuando grandes cantidad de datos deben ser movidos, se requiere una técnica más eficiente: el acceso directo a memoria (DMA).

Función Del DMA

El DMA requiere un módulo adicional en el bus del sistema. El módulo de DMA (Figura 6.21) es capaz de simular al procesador y tomar el control del sistema desde la CPU. La técnica opera de la siguiente manera. Cuando el procesador desea leer o escribir un bloque de datos, emite un comando al módulo de DMA, enviándole al módulo de DMA la siguiente información:

- Si la operación solicitada es una lectura o una escritura.
- La dirección del dispositivo de E/S involucrado.
- La posición inicial en memoria a leer o a escribir.
- El número de palabras a ser leído o escrito.

Transferencia de datos mediante DMA.

Cinco formas de obtener el control del bus:

Por ráfagas: Cuando el DMA toma el control del bus, no lo libera hasta haber transmitido el bloque de datos pedido. Se consigue la mayor velocidad de transferencia, pero se tiene inactiva la CPU (parada del procesador).

Por robo de ciclos: Cuando el DMA toma el control del bus, lo retiene durante un solo ciclo. Transmite una palabra y libera el bus. Es la forma más usual. Reduce al máximo la velocidad de transferencia y la interferencia del DMA sobre la actividad del procesador.

DMA transparente: Se elimina completamente la interferencia entre el DMA y la CPU. Sólo se roban ciclos cuando la cpu no está utilizando el bus del sistema. No se obtiene una velocidad de transferencia muy elevada.

Por demanda: El periférico comienza la transferencia por DMA, pero devuelve el control a la CPU cuando no tienen más datos disponibles. Tan pronto como el periférico tiene nuevos datos, vuelve a retomar el control del bus y se continúa así hasta que acaba la transferencia completa del bloque.

Dato a dato: Cada vez que el periférico solicita una transferencia por DMA, se envía un único dato y se devuelve el control a la CPU. El proceso acaba cuando se ha transferido todo el bloque. De utilidad cuando se desea simultanear la ejecución de un programa con la recepción o transmisión de datos a velocidades moderadas. La CPU sigue ejecutando su programa casi con la misma velocidad mientras se efectúa la transferencia de datos.

Los ciclos de bus utilizados por el DMA no suponen una interrupción. El procesador no guarda su contexto, puesto que el controlador DMA no altera los registros de la CPU. El procesador para su actividad durante un ciclo de bus.

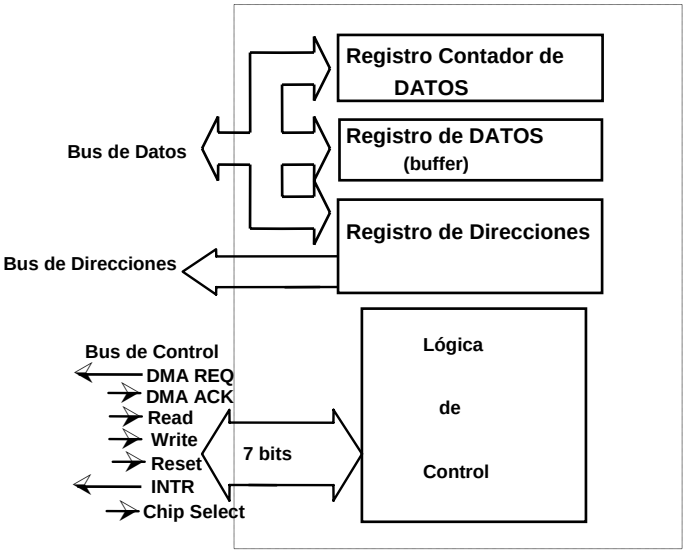
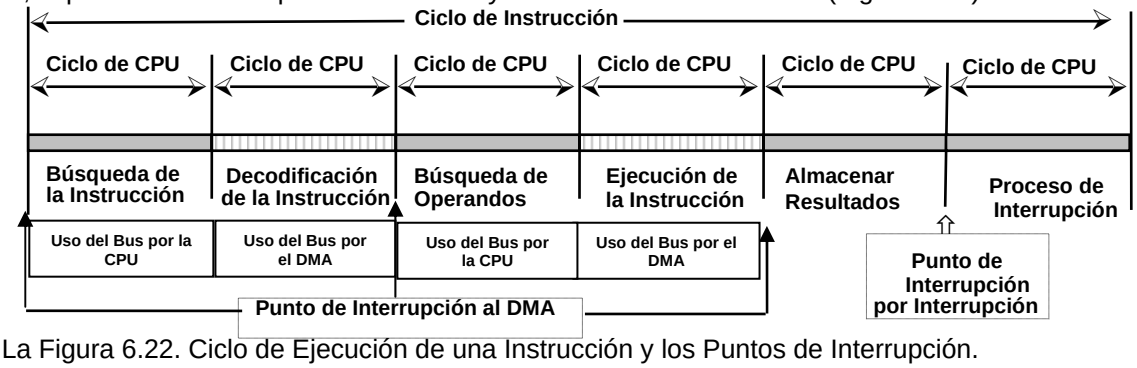


Figura 6.21 - El módulo de Acceso Directo a Memoria (DMA)

El procesador luego continúa con otro trabajo. Ha delegado esta operación de E/S al módulo de DMA, y ese módulo se hará cargo de la operación. El módulo de DMA transfiere el bloque entero de datos, una palabra a la vez, directamente desde o hacia la memoria, sin pasar por el procesador. Cuando la transferencia se ha completado, el módulo de DMA envía una señal de interrupción a la CPU. En consecuencia, el procesador sólo opera al comienzo y al final de la transferencia. (Figura 6.20).



La Figura 6.22. Ciclo de Ejecución de una Instrucción y los Puntos de Interrupción.

El módulo de DMA necesita tomar control del bus del sistema para poder transferir los datos desde y hacia la memoria. Para este propósito, el módulo de DMA debe utilizar el bus sólo cuando el procesador no lo necesita, o debe forzar a que el procesador suspenda temporariamente su operación. Esta última técnica es la más común y se la conoce como ***robo de ciclo*** dado que el módulo de DMA en efecto roba ciclos del bus.

La Figura 6.22 muestra cuando en el ciclo de instrucción el procesador puede ser suspendido. En cada caso, el procesador es suspendido justo antes de que ésta necesita utilizar el bus. El módulo de DMA luego transfiere una palabra y devuelve el control a la CPU. Observe que esto no es una interrupción; el procesador no salva el contexto y hace otra cosa. En vez, el procesador hace una pausa de un ciclo. El efecto general es que el procesador ejecuta las instrucciones algo más despacio. Aún así, el DMA es mucho más eficiente que la E/S programada o por interrupciones.

El mecanismo del DMA puede ser configurado de varias formas. Algunas posibilidades se muestran en la Figura 6.23. En el primer ejemplo, todos los módulos comparten el mismo bus del sistema. El módulo de DMA, actuando como una CPU sustituta, utiliza E/S programada para intercambiar datos entre memoria y el módulo de E/S a través del módulo de DMA. Esta configuración, si bien puede ser barata, claramente

es ineficiente. Al igual que en la E/S programada controlada por la CPU, cada transferencia de una palabra consume 2 ciclos de bus.

El número de ciclos de bus requeridos puede ser substancialmente bajado integrando las funciones de DMA junto a la funciones de E/S. Como se indica en la Figura 6.23-b, esto significa que hay una trayectoria entre el módulo de DMA y uno o más módulos de E/S que no incluye al bus del sistema. La lógica del DMA puede también ser parte del un módulo de E/S, o puede ser un módulo separado que controla uno o más módulos de E/S. Se puede avanzar un paso más en este concepto conectando los módulos de E/S al módulo de DMA a un bus de E/S (Figura 6.23-c). Esto reduce el número de interfases a uno en el módulo de DMA y provee una configuración fácilmente expansible. En todos estos casos (Figuras 6.23-b y 6.23-c), el bus del sistema que el módulo de DMA comparte con el procesador y la memoria es utilizado por el DMA sólo para intercambiar datos con la memoria. El intercambio de datos con los módulos de E/S toma lugar fuera del bus del sistema.

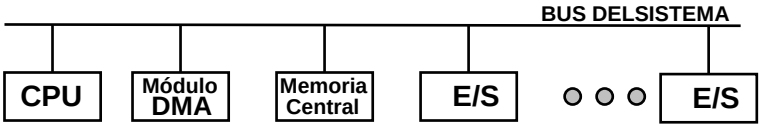


Figura 6.23 - a: DMA conectado simplemente a un BUS

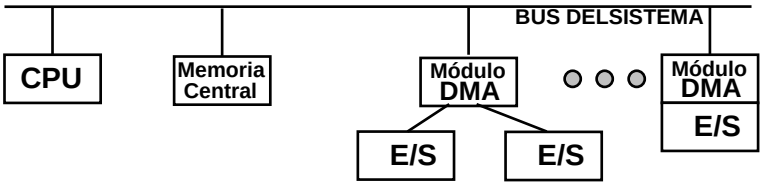


Figura 6.23 - b: DMA integrado al módulo de E/S

conectado simplemente a un

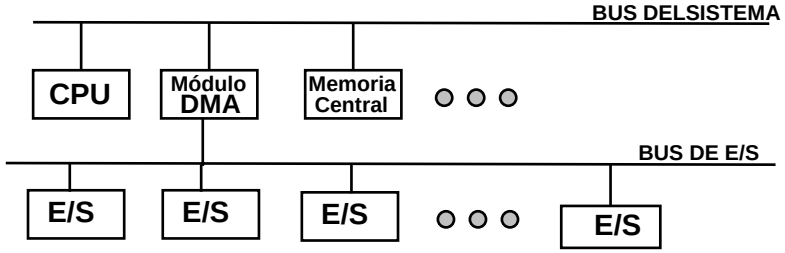


Figura 6.23 - c: DMA conectado a un BUS de E/S y al BUS del Sistema.

Figura 6.23 Configuraciones posibles con DMA

6.5.4. Principios del Software de E/S

La idea básica es organizar el software como una serie de capas, donde la de más abajo se ocupen de esconder las peculiaridades del HARDWARE a las de arriba, y las de mas arriba se ocupen de presentar una interfase agradable, limpia y regular a los usuarios. Para ello deberán proveer un conjunto de niveles de servicios como se indica en la Figura 6.24.

Los niveles que intervienen son:

1. **Procesos de usuario:** requerimiento de E/S. Se arma el Pedido de E/S que es un proceso que arma el S.O. para llevar a cabo el servicio de la E/S (el usuario hace la solicitud y provee otros datos). Este proceso se ubica en una cola hasta tanto se le asigne el periférico.
2. **E/S lógica** (Software, independiente del dispositivo): el módulo de E/S lógica trata al dispositivo como un recurso lógico y no se preocupa de los detalles de control real del dispositivo. Se ocupa de la gestión de funciones generales de E/S pedidas por los procesos de usuario, permitiéndoles manejar el dispositivo mediante un identificador y órdenes simples como Abrir, Cerrar, Leer y Escribir. Se definen los nombres de los periféricos, el manejo del buffering y la asignación del proceso al periférico.
3. **E/S con dispositivos** (device driver): se fija si el periférico está funcionando bien. Si está correcto, las operaciones pedidas y los datos (caracteres almacenados, registros, etc.) se convierten en secuencias adecuadas de instrucciones de E/S, comandos de canal y órdenes al controlador. Si hay error, manda mensaje de error al nivel 2 y luego al nivel 1.
4. **Planificación y Control** (Interrupt handler: manejadores de interrupciones): la planificación y encolado y el control de las operaciones de E/S ocurren en este nivel. Se averigua e informa sobre el estado de la E/S y se manejan las interrupciones. Cuando termina la E/S, se produce

una interrupción del fin de E/S. Este es el nivel del software que realmente interacciona con el módulo de E/S y, por lo tanto, con el hardware del dispositivo.

5. **Hardware**: se encarga de llevar a cabo la E/S.

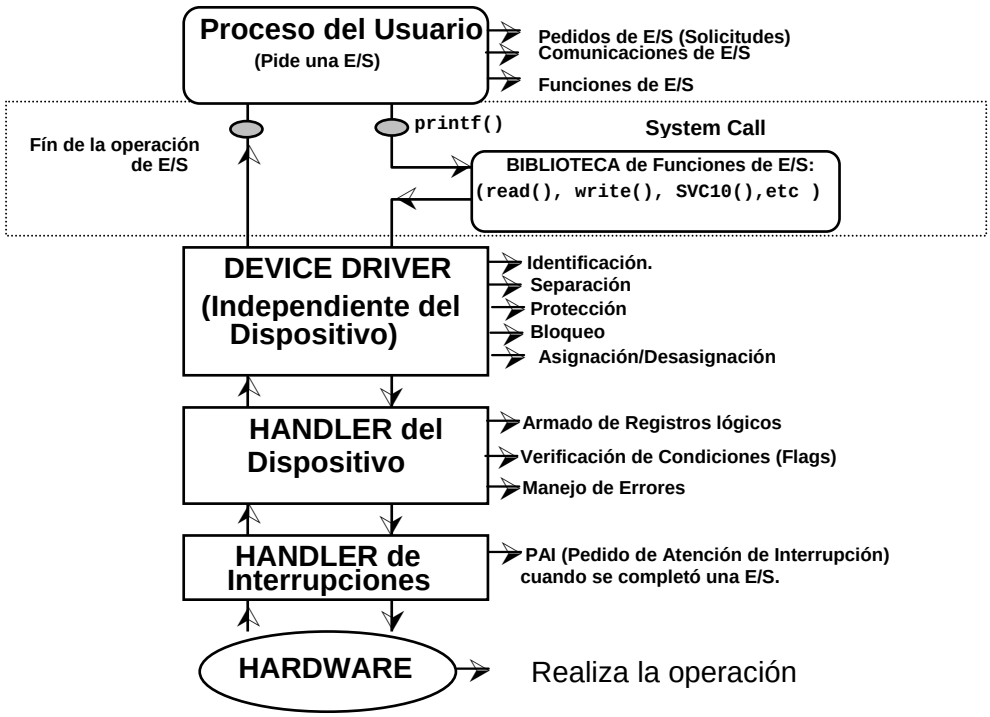


Figura 6.24 Los Niveles de Servicios en una Operación de E/S.

De todos modos, la mayor responsabilidad del Software de E/S es establecer la vinculación entre los distintos módulos para realizar la transferencia de los datos en forma controlada. Para ello utiliza una serie de Tablas y un conjunto de rutinas especializadas que se ocupan de organizar, administrar y operar dicha transferencia de datos. En el caso de máquinas mainframe se establece un Sistema de Control Físico de E/S (PIOCS -Phisical Input Output Control System) y un Sistema de Control Lógico de E/S (LIOCS -Logical Input Output Control System).

Un requisito que el software debe cumplir obligatoriamente es el de independencia de los dispositivos, o sea que el software de E/S debe ser escrito de tal manera que pueda ejecutar satisfactoriamente sin importar el periférico sobre el que se opera.

Esto último se logra tratando a los periféricos de una manera uniforme. Cada periférico tiene una representación interna de los códigos estándares, es por eso que se deberá realizar una traducción para evitar problemas. Otro punto importante es que los programas deberán trabajar sobre periféricos virtuales (streams en atlas o archivos en MULTICS) sin hacer referencia a los periféricos físicos.

En este contexto los drivers de los dispositivos (Device Drivers) de entrada/salida juegan un papel fundamental ya que son los encargados de ejecutar y controlar las operaciones de E/S realizadas sobre el periférico que manejan. Un driver está formado por tablas y rutinas que sirven de interfase entre el sistema operativo y el periférico bajo su control.

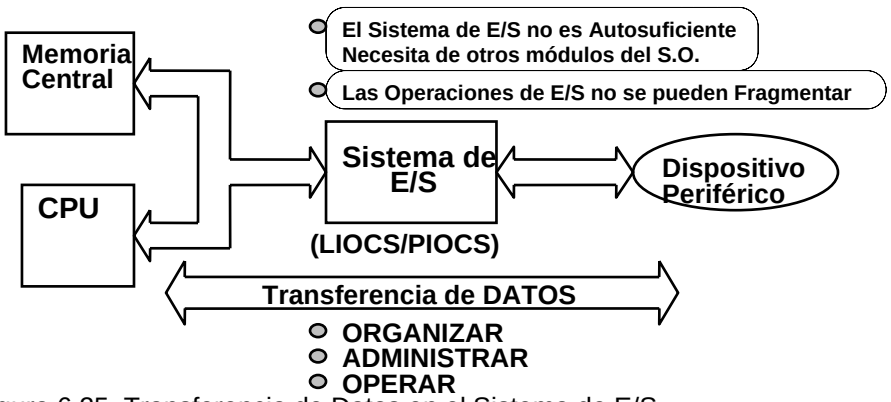


Figura 6.25. Transferencia de Datos en el Sistema de E/S

Cada driver tiene un solo tipo de dispositivo de E/S (o varios de la misma clase), y traduce los pedidos del S.O. a un *lenguaje* entendible para los periféricos.

Las principales razones del porqué el S.O. debe controlar las operaciones de Entrada / Salida son:

- Las interfases del Hardware requieren de un Software complejo para controlarlo y usarlo.
- Los Dispositivos Periféricos son recursos compartidos y
- El S.O. provee una interfase consistente, uniforme y flexible para todos los Dispositivos.

Para dicha implementación, debemos reproducir exactamente la misma información, en más de un almacenamiento de memoria no volátil (Ejemplo Buffers, área de datos de memoria central, etc.). Además necesitamos actualizar la información de una forma controlada para asegurar que las fallas que pudieran ocurrir durante la transferencia, no dañen la información necesaria.

Una transferencia entre la memoria y el disco puede tener uno de los siguientes resultados:

a) Terminación exitosa: la información transferida llega segura a su destino (Ver Figura 6.27).
b) Falla parcial: la falla ocurre en la mitad de la transferencia y el bloque de destino tiene información errónea.
c) Falla total: la falla ocurre lo suficientemente temprano durante la transferencia, de modo que el bloque destino queda intacto.

Interrupción de la E/S: Se produce una interrupción voluntaria o no para que se no se complete la operación en curso.

Si ocurre una falla, el sistema debe restablecer el bloque en un estado consistente, para esto debe tener dos bloques físicos por cada bloque lógico.

Durante la recuperación, luego de una falla, cada par de bloques físicos es examinado por el Hardware y Software. Si ambos son iguales y no existe un error detectable, entonces no se requieren acciones adicionales. Si un bloque contiene un error detectable, se reemplaza su contenido con el valor del segundo bloque. Si ambos contienen un error indetectable, pero ellos son distintos en contenido, entonces se reemplaza en valor del primero, con el valor del segundo.

Se pueden agregar copias, en este procedimiento, para reducir la probabilidad de fallas.

El Proceso de Inicialización de la Entrada / Salida (I/O Bootstrap)

La inicialización de la E/S es parte del proceso general de inicialización (“Booteo” o Cold start) de la máquina que realiza el S.O. en que prepara el hardware y su software para poder cargar el S.O. y para ello crea tablas de Drivers, de dispositivos e inicializa las mismas, vinculándolas (Binding entre dispositivos, Controladores, al igual que los puertos asociados, canales, buses, interfases, etc.). Esto ocurre después de haber encendido la máquina en el arranque en frío (Cold Start) en que se produce el RESET del Sistema y se inicia la carga del S.O. generándose los 4 pasos indicados en la Figura 6.26.

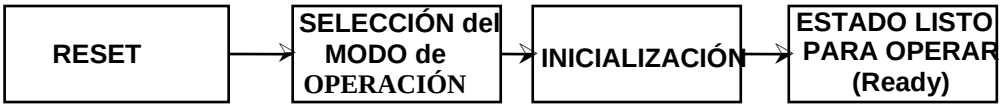


Figura 6.26 Proceso de Inicialización de los módulos de E / S

Para cada paso se producen las siguientes acciones:

- RESET:

1. Estado conocido.
2. Limpia los Buffers de Datos.
3. Limpia los Registros de Estado y Órdenes de cada controlador.
4. Habilita el Registro de Ordenes para recibir la ORDEN DE MODO.

- SELECCIÓN DEL MODO DE OPERACIÓN

5. Transferencia de la orden de modo de funcionamiento (asincrónico o sincrónico, paridad par o impar, Cantidad de bits: 6, 7, 8, o 9, etc.)
6. Habilitación para la transmisión - Recepción mediante el Registro de órdenes (Tx_hab o Rx_hab)
7. Verificación del funcionamiento de las Rutinas de errores.

- INICIALIZACIÓN:

8. Inicializar el Port y el Device Driver y preparar TABLAS.
9. Especificar el “Modo de Operación”
10. Especificar las características de Dispositivo-Controlador.
11. Carga de Programas y rutinas (del Driver, editores, bloqueadores, interrupciones, inicio de operaciones, etc.)
12. Construye en Memoria Central las TABLAS (I/O Address en I/O Page)
13. Vincular (BINDING) cargando los punteros en las respectivas TABLAS.

- ESTADO DE LISTO PARA OPERAR:

14. Si todos los pasos anteriores dieron bien, se carga el Registro de Estado con “READY” quedando disponible para la normal operación del periférico ante cualquier requerimiento, sobre todo, a partir de aquí se carga el S. O. en memoria central.

6.5.5. Metas del Software de E/S

Un concepto clave en el diseño de software de E/S es la **independencia de los dispositivos**. Debería escribirse los programas en forma tal de que no sea dependiente de un periférico para su ejecución, sino que puedan usarse tanto desde un disco duro como de una disquetera o una cinta, sin tener que recompilarlos. El S.O. debería encargarse de los problemas causados por el hecho de que los dispositivos son diferentes y que necesitan drivers totalmente distintos.

Otra meta es la **uniformidad de los nombres**. Los nombres deberían ser simplemente una cadena o un entero y no deberían depender del dispositivo.

Otro problema importante del S.O. es **la manipulación de errores**. En general los errores deberían tratarse tan cerca del HARDWARE como sea posible. Si un controlador encuentra un error debería corregirlo el mismo.

Otro concepto clave es **transferencia** SINCRÓNICA (bloqueada) y la transferencia ASINCRÓNICA (llevada por interrupciones). La mayoría de la E/S física es sincrónica - el procesador comienza la transferencia de datos y va a hacer otra cosa hasta que termina.

El último concepto son los **dispositivos compartibles y los dedicados**. Algunos dispositivos (ej.: discos) se pueden compartir entre varios usuarios al mismo tiempo. Otros dispositivos (ej.: impresoras) se tienen que dedicar a un solo usuario por vez. El tener dispositivos dedicados trae muchos problemas, estos problemas los tiene que manipular el S.O..

Estos objetivos se pueden alcanzar de una forma coherente y eficiente estructurando el software de E/S en los siguientes estratos:

- Software de nivel del usuario
- Software de sistema independientes de los dispositivos
- Drivers de dispositivos
- Manejadores de interrupciones

La Fig. 6.24 indica estos niveles.

6.5.6.- Manejadores de Interrupciones (Interrupt handler)

Las interrupciones deben ocultarse, tratando de que la menor parte posible del sistema se entere de que ocurren. La mejor manera de ocultarlas es tener todos los procesos que comienzan una E/S bloqueados hasta que terminen y se haya producido una interrupción. El proceso puede autobloquearse ejecutando una operación P(s) o un DOWN() en un semáforo, un WAIT() en una variable condicional, o un RECEIVE() en un mensaje, por ejemplo. Cuando ocurre una interrupción, el procedimiento de la interrupción hace lo que tenga que hacer para desbloquear el proceso que la generó. Solamente debe generar un PAI (Pedido de Atención de Interrupción) al procesador (First Level Interrupt Handler) cuando se ha completado la operación de E/S, o se produjo un error que no pudo ser resuelto en su nivel. Algunos errores comunes que se presentan son:

- error de programación en la E/S.
- error de checksum transitorio
- error de checksum permanente
- error de búsqueda
- error de controlador

El driver debe manejar estos errores.

6.5.7.- Drivers de Dispositivos

Como definición, podemos decir que un Driver **(device driver)** es el software formado por un conjunto de rutinas y tablas que, instalados, forman parte del S.O. y sirven para ejecutar y controlar todas las operaciones de E/S que se realizan sobre el periférico conectado a la computadora y que controla dicho Driver. Es un conjunto de programas que provee la interfase entre el Sistema Operativo y un dado tipo de dispositivo periférico.

Todos los códigos, que dependen del dispositivo, están en el driver. Cada driver maneja un tipo de dispositivo, o al menos, una clase de dispositivos muy similares. Cada controlador tiene uno o más registros

que se usan para darle órdenes o comandos. Es el software que traslada los comandos en términos entendibles por el periférico y el S.O. y actúa como interfase entre ellos.

La tarea que desempeña un driver de algún dispositivo de E/S se reduce a aceptar pedidos que realizan los programas que se están ejecutando y garantizar la ejecución de las rutinas de E/S correspondientes.

Funciones del Device Driver:

Las principales funciones que realiza un Device Driver son:

- Definir las características Lógicas - Físicas del dispositivo que controla.
- Inicializar los Registros y el modo de funcionamiento en el momento de arranque de la máquina (booteo).
- Habilitar y deshabilitar el dispositivo para un dado proceso.
- Controlar los accesos según los permisos otorgados a cada usuario.
- Permitir la ejecución concurrente de operaciones de E/S en forma controlada.
- Identificar y Procesar todas las operaciones de E/S definidas para los usuarios o programas a través de llamadas al sistema (SYSCALL).
- Cancelar las operaciones de E/S sobre el dispositivo cuando se produzca algún evento que así lo requiera.
- Procesar todas las interrupciones del Hardware de E/S o generada por el dispositivo que controla el Driver.
- Tratar todos los errores y el estado del dispositivo haciendo la correspondiente comunicación al usuario o el programa que requirió el servicio de E/S.
- Construir el programa canal en los equipos mainframe.
- Separar la complejidad operativa de cada dispositivo brindando una interfase homogénea y abstracta al S.O. del mismo.
- Vincular y desvincular los módulos que intervienen en la operación de E/S.
- Bloquear y desbloquear los datos durante la transferencia.

En términos generales, el trabajo de un driver es aceptar pedidos abstractos del software independiente del dispositivo que esta por encima de él, y controlar que se ejecute. Cuando se pide ejecutar una acción y el dispositivo esta inactivo, inmediatamente lo pone activo y ejecuta la operación solicitada, en cambio si está ocupado, pondrá el pedido en una cola de espera para realizar la operación mas tarde. El primer paso para llevar a cabo un pedido de E/S es traducirlo a términos concretos, como se ve en la figura 6.27.

Una vez que el S.O. determinó que comandos debe realizar al controlador, se comienza a enviar las operaciones, escribiendo en el registro de órdenes del controlador. Algunos controladores solo pueden manejar un comando por vez. Otros pueden aceptar una lista enlazada o vinculada (linkeada) de comandos, que pueden realizarse solos sin la ayuda del S.O. Después que estos comandos fueron enviados, se puede aplicar una de las dos situaciones siguientes:

- En la mayoría de los casos el driver debe esperar hasta que el controlador haga algún trabajo, entonces se autobloquea hasta que se produzca una interrupción que lo desbloquee.
- En otros casos, la operación termina sin ninguna demora, entonces no se necesita bloquear el driver.

En ambos casos, luego de que la operación se completo se debe verificar el resultado de como se ha completado la operación. Si todo esta bien, el driver puede tener datos para pasar al software independiente. Finalmente devuelve información de STATUS al que lo llamó. Si hay otros pedidos encolados, ya se puede seleccionar uno para comenzarse, sino el driver se bloquea hasta recibir otro pedido.

Tipos de Device Driver

Normalmente los drivers pueden ser clasificados como drivers orientados a bloques y drivers orientados a caracteres. La diferencia radica en la cantidad de caracteres que pueden procesar por cada operación que realizan.

Los drivers orientados a bloques manejan, como su nombre lo indica, bloques de caracteres de n bytes. Los orientados a caracteres operarán entonces sobre un único carácter cada vez que realicen una operación. Los orientados a caracteres pueden también manejar los caracteres especiales o no según sea necesario en cada caso.

Entonces, Existen dos tipos de device driver:

- Block Device Driver
- Character Device Driver

Los orientados a Bloques controlan los periféricos con accesos basados en una dirección de datos compuestos por varios caracteres, o sea que, manipulan varios bytes en una misma operación. Estos Device Drivers pueden soportar y controlar más de una unidad de dispositivos idénticos. Como ejemplo están los minidiscos y las disqueteras.

Los Device Drivers orientados a caracteres controlan aquellos periféricos que transmiten o reciben un solo caracter por vez en cada operación. Las impresoras y el módem son ejemplos de estos dispositivos. Estos últimos pueden manejar la información según uno de los siguientes modos:

- mode raw, procesa cada byte sin atender los caracteres especiales (Ejemplo el enter de un teclado)
- mode cooked, que sí presta atención a los caracteres especiales.

Puntos a tener en cuenta en el desarrollo de drivers:

En los S.O. hay que tener en cuenta algunas consideraciones extras al momento de crear un driver, ya que este paquete puede proporcionar ventajas como el multiprocesamiento y la recuperación ante fallas eléctricas. Si se va a utilizar las características de multiprocesamiento es necesario proteger los recursos compartidos para que no sean utilizados por dos hilos de ejecución al mismo tiempo. De la misma manera que otros componentes, un driver puede ejecutarse en dos o más procesadores a la vez, para lo cual es necesario añadir algunas herramientas que permitan la sincronización.

En cuanto a las fallas por cortes de energía, se puede activar una opción a los drivers, siempre y cuando el sistema tenga una UPS[4]; esta opción es conocida como iniciación o arranque en caliente; es decir, al suceder un corte de energía el usuario activa la opción que permite cerrar el sistema tal como se encuentra en ese momento, gracias a la energía suministrada por la UPS; al reestablecerse la energía de nuevo y encender el sistema, continúan ejecutándose los procesos y tareas que se encontraban en el procesador antes del corte de energía.

Manejo de errores.

Parte del manejo de errores la realiza el subsistema de I/O. El manejo de errores se hace principalmente en el driver, pues la mayoría de los errores son dependientes del dispositivo (y por ende, sólo el driver podría saber qué hacer, sobre todo si el error es transitorio). Por ejemplo, si el driver recibe un mensaje de error del controlador al intentar leer un bloque, reintentará varias veces antes de reportar el error a la capa superior. Entonces el subsistema de I/O decide qué hacer, en una forma independiente del dispositivo.

6.5.8. Pasos y Controles en una operación de E/S .

Todos los Pedidos de Entrada / Salida (PES) que efectúan los procesos de los usuarios, se realizan mediante una llamada al Sistema Operativo (SYSCALL), quién brindará el servicio de acuerdo al tipo de operación solicitada. A partir de ésta acción se generan los siguientes pasos (ver figura 6.27):

1. El Proceso del Usuario solicita un servicio (en la figura 6.27 pide leer el archivo TOTO). Para ello invoca una llamada al sistema SYSCALL leer “TOTO”.
2. El Kernel inmediatamente ejecuta un “context switch” salvando el Bloque de Control del Proceso (PCB) que solicitó el servicio colocando su Estado en Bloqueado por E/S (Status = Wait for I/O)
3. Luego invoca la rutina de servicio del SYSCALL, deduce los parámetros, verifica su correspondencia y los permisos de acceso para dicho servicio.
4. De acuerdo a este control, la llamada al sistema puede ser rechazada o no. Si es rechazada, se coloca el error en la Palabra de estado (Status word) del proceso que solicitó el servicio y se devuelve el control para que él o el S.O. lo traten. Si los parámetros y los permisos son correctos, la llamada del sistema construye una estructura de datos que denominamos “Pedido de Entrada / Salida o PES” con la cual se vincula lógicamente y físicamente el Driver (Bloque de Control del Driver BCD) y la Unidad (Bloque de Control de la Unidad) o dispositivo. Generando el camino (path) de vinculación involucrada en la operación de E/S solicitada.
5. Para determinar si es factible efectuar la operación de E/S solicitada, el S.O. lee el registro de estado del controlador que puede estar libre u ocupado. Si está ocupado coloca al PES en la cola de PES en base a la política de ordenamiento de la cola (Ejemplo Planificación de brazo o First Come, First Served)
6. Si el Driver está libre lo ocupa y coloca en estado “ocupado y el PES como “en servicio”. Activa el “Time out” que controla el tiempo de proceso de la E/S, invoca la rutina de iniciar la operación de E/S para lo cual el S.O. carga en el registro del controlador la operación solicitada (Leer o Escribir). A partir de ésta acción el S.O. finaliza su llamada SYSCALL y si el procesador es multiprogramada producirá un cambio

[4] UPS = Unintrrupt Power System (sistema de alimentación eléctrica no interrumpible)

de contexto ejecutando otro u otros procesos de la cola de listos mientras dure la E/S como se indica en la Figura 6.27. Si el procesador es monoprogramada quedará en espera por la interrupción de E/S completada.

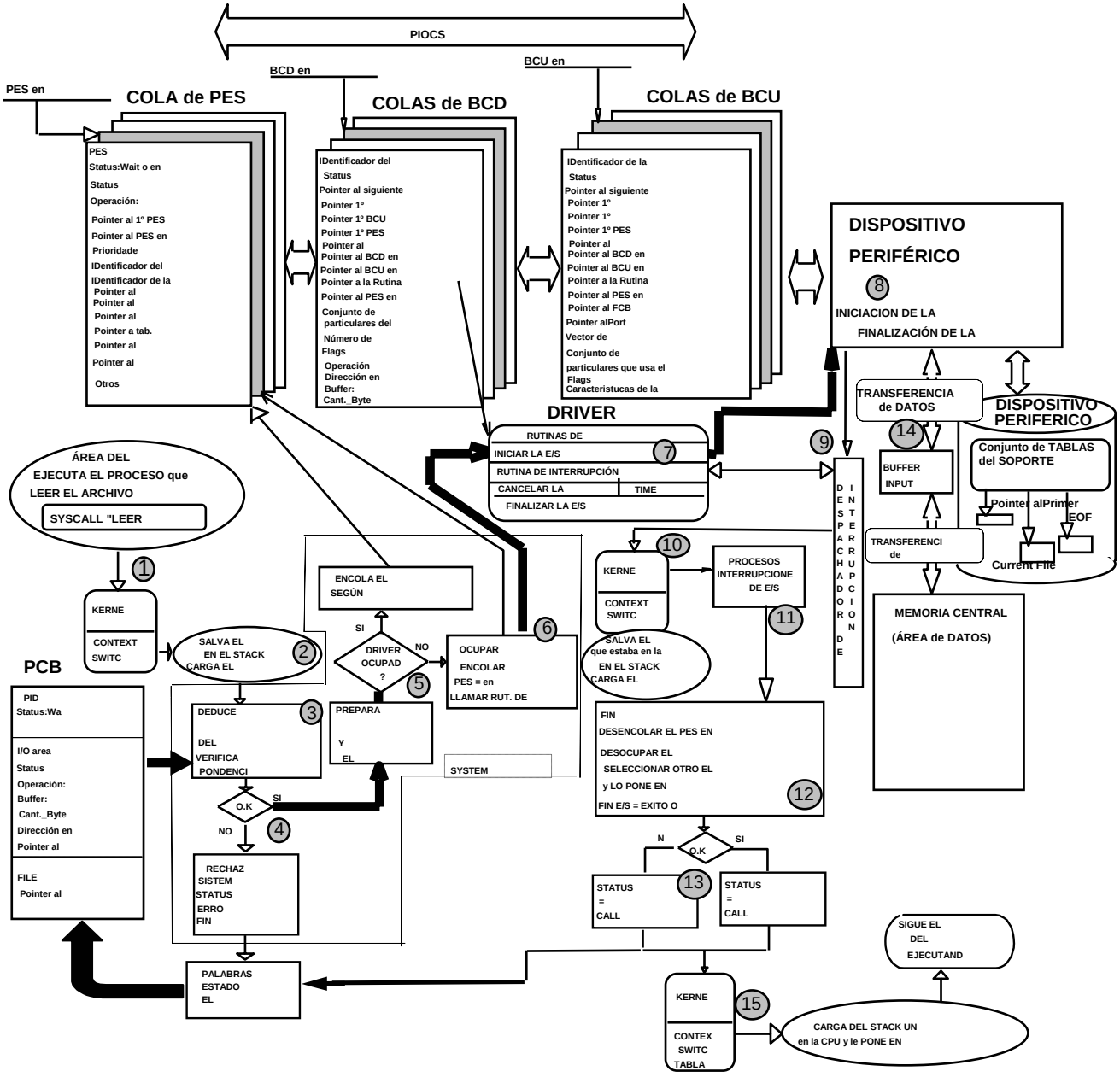


Figura 6.27 - Pasos y Controles en una Operación de E/S (Lectura/Escritura) en DISCO

7. En forma totalmente independiente del S.O., el Driver inicia la operación solicitada, habilitando el controlador y el dispositivo periférico correspondiente. Las respectivas estructuras de datos (BCD y BCU), que fueran creadas, habilitadas y vinculadas durante el proceso de inicialización de la máquina (Booteo). Así queda establecido el camino (path) lógico y físico entre el puerto, controlador y dispositivo. En caso de que se decida cancelar el PES en servicio se activa la rutina respectiva.
8. El periférico, cuando completa la operación de E/S solicitada, genera una interrupción del dispositivo (algunos dispositivos no disponen de este mecanismo) .
9. El Handler o despachador de interrupciones reconoce el tipo de interrupción ocurrida, activa o inhibe la interrupción e invoca la rutina de servicio de la interrupción respectiva generando
10. un PAI (Pedido de Atención de Interrupción) o INTR (interrupt request) para que el procesador lo atienda. El PAI genera que el Kernel realice un “context switch”,
11. El Kernel con el “context switch” apila en el Stack el PCB del proceso que estaba ejecutando y carga la rutina de servicio que solicitó el PAI.

Como consecuencia de la finalización de la operación de E/S involucrada en el PES, se ejecutan las siguientes acciones:

- Desencolar el PES que estaba en Servicio
- Detectar errores en la transferencia de datos y la ejecución y colocar el resultado (Éxito o Fracaso) en el Registro de estado.
- Seleccionar otro PES (si es que la cola no está vacía) y colocarlo "en Servicio".
- Iniciar la ejecución desde el punto 7 en adelante.

Si la cola está vacía, poner el estado del driver en "desocupado o libre (Ready).

12. De acuerdo al resultado de la operación (Éxito o Fracaso), se coloca la palabra de estado para el proceso del usuario que requirió la operación de E/S y se inicia la transferencia de datos controlada, ya sea por DMA o por CPU como ya se explicó.
13. Transferencia de datos entre periférico y el área de Datos de la memoria central
14. El Kernel realiza nuevamente un context switch cargando un nuevo proceso y lo pone en ejecución. Puede ser el mismo proceso que solicitó el PES si su prioridad lo justifica. De esta forma continúa la normal ejecución de los Procesos.

En la Figura 6.27 se observan distintas colas formadas por las estructuras de datos que controlan los distintos Drivers (BLOQUE DE CONTROL DE DRIVERS - BCD, BLOQUE DE CONTROL DE UNIDAD - BCU) y de dispositivo, además de la Cola de Pedidos de E/S (PES)**.** Cada dispositivo tiene su propia cola de PES. Si es un dispositivo no compartible solo tendrá un proceso activo o en servicio (como en el caso de cinta o impresora), sino puede haber varios procesos (como en disco) que comparten el recurso.

6.5.9.- Software de E/S Independiente del Dispositivo

A pesar de que algunos softwares de E/S son específicos del dispositivo, una gran parte del software de E/S es INDEPENDIENTE. La diferencia exacta entre drivers y software de E/S independiente es el sistema al que están subordinados, porque algunas funciones que se podrían hacer de una manera independiente del dispositivo también se podrían hacer en el driver, por razones de eficiencia u otras razones. La función básica del software independiente es hacer funciones de E/S que son comunes a todos los dispositivos, y que proveen una interfase uniforme al software del nivel de usuario. Un punto importante en un S.O. es como se denominan los objetos en el sistema. El software independiente se encarga de mapear los nombres de dispositivos simbólicos y transformarlos en los nombres correctos del driver. Muy relacionado con este tema, está la protección. En algunos Sistemas Operativos de computadoras personales (como el MS-DOS®, no dispone de protección alguna. En la mayoría de los sistemas de Mainframes, que un usuario acceda a los dispositivos de E/S está totalmente prohibido. En UNIX, se usa un método un poco más flexible. Los archivos especiales que corresponden a los dispositivos de E/S están protegidos por los bits RWX (read, write, execute). El administrador del sistema puede colocar los permisos correctos para cada dispositivo.

Otro punto es el buffering, tanto para los dispositivos de caracteres como para los de bloque. Algunos dispositivos pueden ser usados solo por un proceso a la vez. El manejo de los errores lo hace generalmente el driver. La mayoría de los errores son provocados por el dispositivo, por lo tanto solo el driver sabe que hacer con estos errores.

6.5.10.- Software de E/S del Espacio del Usuario

A pesar de que la mayoría del software de E/S está dentro del S.O., una pequeña porción del software de E/S consiste en Bibliotecas vinculadas (linkeditadas) - nótese que se dijo **biblioteca (library**)- **y no librería (Book store)** junto con los programas del usuario, (por ejemplo las llamadas al sistema).

Hay algunos procedimientos que lo único que hacen es darle los parámetros y llamar a una rutina del Sistema Operativo. También hay otros que ejecutan un procedimiento, transforman los datos y luego hacen la llamada al sistema (ej.: printf ()).

No todo el software de E/S del nivel del usuario, consiste en un conjunto de Bibliotecas. Otra categoría importante de software es el sistema de SPOOLING (maneja dispositivos dedicados en un sistema multiprogramado).

SPOOLING se trata de simular un periférico en línea (Simultaneus Peripherical Operation On Line) mientras no hay disponibilidad de los mismos: Si un proceso solicita una impresora y ésta impresora le es asignado al proceso. Este proceso puede tardar demasiado tiempo en procesar los datos para luego imprimirlos, por lo tanto tiene la impresora asignada sin utilizarla, y no permite que otro proceso la pueda usar. El SPOOLING crea un proceso especial, que se llama DAEMON[5] (Servidor de Impresión), y un

[5] El concepto de Daemon es el de un proceso servidor que pertenece al S.O.

directorio especial que se llama directorio de SPOOLING. Para imprimir, el proceso genera primero un archivo con los datos que desea imprimir y lo pone en el directorio de spooling. El daemon, que es el único que puede imprimir, imprime los archivos que se encuentran en el directorio de spooling.

6.5.11.- Software de Entrada

El teclado y el monitor son casi dispositivos independientes entre sí (no son totalmente independientes, porque cada caracter que se tipee en el Teclado se debe escribir en el monitor).

El trabajo básico del driver del teclado es juntar la información desde el teclado y pasarla al programa del usuario cuando lo lee de la terminal. El driver puede tomar dos filosofías. En la primera, el trabajo del driver es solo ingresar el input y pasarlo sin ninguna modificación. Esta filosofa es muy útil para los editores de pantalla muy sofisticados. La mayoría de los programas no quieren demasiados detalles. Solo quieren el input correcto, no la secuencia exacta que se tipeó (por ejemplo: si corrijo, no quiero que envíe los retrocesos de carro que escribí para corregir). Esta observación nos lleva a una segunda filosofía: el driver maneja toda la edición de la entrelinea, y que envía toda la línea corregida. La primer filosofía es la orientada a caracteres, la segunda orientada a la línea (modo RAW, y modo COOKED). Muchos sistemas proveen ambas, dentro de la cuales hay que seleccionar una.

Una vez que el driver recibió el caracter, debe comenzar a procesarlo. Transformando el número de tecla en un caracter ASCII.

Si la terminal esta en modo COOK, los caracteres se deben almacenar en un buffer hasta que la línea se complete. Aunque la terminal este en modo RAW, puede que el input todavía no se haya pedido, por lo tanto también debe almacenar los caracteres en un buffer.

Hay dos métodos para este buffering. En el primero, el driver contiene un conjunto central de registros buffers, cuya capacidad es fija por ejemplo 16 caracteres. Una estructura de datos (Tabla) esta asociada con cada terminal, que contiene entre, otros datos, un puntero a la cadena de buffers para colectar la entrada de esa terminal. Cuando se ingresan más caracteres, se adicionan, (automáticamente en algunos S.O.) buffers a la cadena. Cuando los caracteres se pasan al programa del usuario esos buffers se devuelven al conjunto del driver.

El otro, es hacer el buffering directamente en la estructura de datos de la terminal misma, sin ningún conjunto de buffers. Como es común que un usuario tipee un comando que tardará muy poco y luego tipee unas pocas líneas a continuación, para que esto sea seguro, el driver debe asignar una dada cantidad, por ejemplo, 200 caracteres por terminal.

El teclado y el monitor son lógicamente dos dispositivos separados. Algunas terminales obligan (en Hardware) a que todo lo que se ingrese por el teclado automáticamente se muestre por pantalla. Muchos Sistemas tienen terminales que no muestran nada cuando se tipea. Por lo tanto queda en manos del software mostrar el input. Esto se llama ECHOING.

El ECHOING es complicado, porque el programa puede estar escribiendo mientras el usuario esta tipeando. Al menos el driver deberá encargarse de buscar un lugar para almacenar el input para que no lo borre el output del programa. También es complicado cuando se tipean más de 80 caracteres en una terminal que tiene 80 caracteres por línea. Algunos drivers truncan las líneas cuando llegan a 80.

Otro problema es el manejo de las tabulaciones (con la tecla TAB) Es un problema del driver saber donde esta el cursor posicionado.

Cuando un S.O. está en modo COOKED, algunos caracteres tienen un significado especial. Es otra tarea del driver saber el significado en cada modo.

Para permitir que los programas especifiquen si quieren trabajar en modo RAW o COOKED, cada S.O. provee una instrucción especial para ello.

6.5.12.- Software de Salida

El método que se usa comúnmente para las terminales RS-232 es tener buffers de salida asociados a cada terminal. Los buffers pueden provenir de un mismo conjunto, como los de entrada, o pueden ser dedicados. Cuando un programa escribe una salida, ésta se copia primero al buffer, y luego también se copia la salida para el ECHOING. Una vez que toda la salida se copió al buffer (o los buffers están llenos), se envía el primer caracter y el driver pone su estado en SLEEP hasta que se produzca una interrupción, entonces envía otro, y así sucesivamente.

Con las terminales de memorias mapeadas, se puede usar un método más simple. Los caracteres que se quieren imprimir se toman de a uno a la vez del espacio del usuario y se colocan directamente en la Memoria de Vídeo (VÍDEO RAM). Con las terminales de memoria mapeadas algunos caracteres necesitan un tratamiento especial (ej.: backspace, retorno de carro). Un driver para memoria mapeada debe conocer con precisión la posición del cursor dentro del software en la VÍDEO RAM, para que los caracteres a imprimir sean puesto en la posición actual del cursor + uno.

En particular, cuando una línea se completa en la parte derecha de la pantalla, el cursor se debe correr a la próxima línea. Generalmente para hacer esto se usa una ayuda especial que el hardware provee. La mayoría de los controladores de vídeo tienen un registro que determina desde donde debe comenzar a enviar bytes desde la VÍDEO RAM. La otra cosa que debe hacer el driver es copiar todo lo que se necesite en la línea siguiente.

Otra cosa que debe hacer el driver de la terminal es determinar la posición actual del cursor. Para esto también el hardware provee algo de ayuda, tiene un registro que almacena la posición a la que el cursor debe ir. Los editores de pantalla y muchos otros programas sofisticados necesitan poder actualizar la pantalla de una forma más compleja.

6.5.13. Procesadores de E/S y Canales de E/S

La evolución de la función de E/S

A medida que los computadores han evolucionado, se observó un patrón creciente en cuanto a la complejidad y sofisticación de las componentes individuales. En ninguna otra parte esto es más evidente que en la función de E/S.

Veremos parte de esta evolución. Los pasos pueden ser resumidos de la siguiente forma:

1. El procesador controla directamente el dispositivo periférico. Esto se observa en dispositivos sencillos controlados por un microprocesador.
2. Se agrega un controlador o un módulo de E/S. El procesador utiliza E/S programada sin interrupciones. Con este paso, el procesador de alguna forma se divorcia de los detalles específicos de las interfases con los dispositivos externos.
3. Se utiliza la misma configuración del paso 2, pero esta vez se emplean interrupciones. El procesador no necesita perder tiempo esperando que una operación de E/S sea ejecutada, incrementando entonces su eficiencia.
4. Se les da acceso directo a memoria a los módulos a través del DMA. Se puede mover ahora un bloque de datos desde o hacia la memoria sin involucrar a la CPU, excepto al comienzo y al final de la transferencia.
5. Es mejorado el módulo de E/S hasta convertirse en un procesador en su propio derecho, con un conjunto especializado de instrucciones de E/S. El procesador ordena al procesador de E/S a ejecutar un programa de E/S residente en memoria. El procesador de E/S obtiene (fetch) y ejecuta estas instrucciones sin intervención de la CPU. Esto permite al procesador especificar una secuencia de actividades de E/S y ser interrumpida sólo cuándo la secuencia entera haya sido procesada.
6. Los módulos de E/S tienen su propia memoria local y esto, de hecho, es un computador sin ir más lejos. Con esta arquitectura, un gran conjunto de dispositivos de E/S pueden ser controlados, con una intervención mínima de la CPU. Un uso común de esta arquitectura se utiliza para controlar un número considerable de terminales interactivas. El procesador de E/S tiene a su cargo la mayoría de las tareas afines para el control de las terminales.

A medida que uno recorre ésta evolución, más y más funciones de E/S son ejecutadas sin la intervención de la CPU. De forma creciente, el procesador es relegada de la funciones de E/S, mejorando la perfomance. En los dos últimos pasos (5 y 6), un cambio mayor ocurre con la introducción del concepto de un módulo de E/S capaz de ejecutar un programa. En el caso del paso 5, un módulo de E/S es a menudo referido como un *canal de E/S.* Para el paso 6, el término generalmente utilizado es *procesador de E/S.* Sin embargo, ambas denominaciones se aplican en ambas situaciones. En lo que sigue utilizaremos el término *canal de E/S.*

Características de los Canales de E/S:

El canal de E/S representa una extensión del concepto de DMA. Un canal de E/S tiene la habilidad de ejecutar instrucciones de E/S, lo que le da un completo control sobre las operaciones de E/S. En un sistema de computación con tales dispositivos, el procesador no ejecuta instrucciones de E/S. Tales instrucciones son almacenadas en la memoria central para ser ejecutadas por un procesador de propósito especial situado en el canal de E/S. Luego el procesador inicia una transferencia de E/S instruyendo al canal de E/S a ejecutar un programa en memoria. El programa especificará el dispositivo o los dispositivos, el área o las áreas de memoria para almacenamiento, prioridades y las acciones que deben tomarse para ciertas condiciones de error. El canal de E/S sigue éstas instrucciones y controla la transferencia de datos.

Dos tipos de canales de E/S son comunes, como se ilustra en la Figura 6.28. Un *canal selector* controla múltiples dispositivos de alta velocidad y, es dedicado a la transferencia de datos con uno de esos dispositivos a la vez. En consecuencia, el canal de E/S selecciona un dispositivo y efectúa la transferencia. Cada dispositivo, o pequeño conjunto de dispositivos, es manejado por un *controlador*, o módulo de E/S,

bastante parecido a los módulos que hemos discutido anteriormente. Luego el canal de E/S sirve en lugar del procesador para controlar a estos controladores de E/S. Un *canal multiplexor* puede manejar E/S con múltiples dispositivos a la vez. Para dispositivos de baja velocidad, un *multiplexor de bytes* acepta o transmite caracteres tan rápido como le sea posible a múltiples dispositivos. Por ejemplo, el flujo de caracteres resultante de tres dispositivos con velocidades distintas y flujos de caracteres individuales $A_1A_2A_3A_4$ ..., $B_1B_2B_3B_4$ ..., y $C_1C_2C_3C_4$ podría ser $A_1B_1C_1A_2C_2A_3B_2C_3A_4$, y así sucesivamente. Para dispositivos de alta velocidad, un *multiplexor de bloques* intercala bloques de datos provenientes de varios dispositivos.

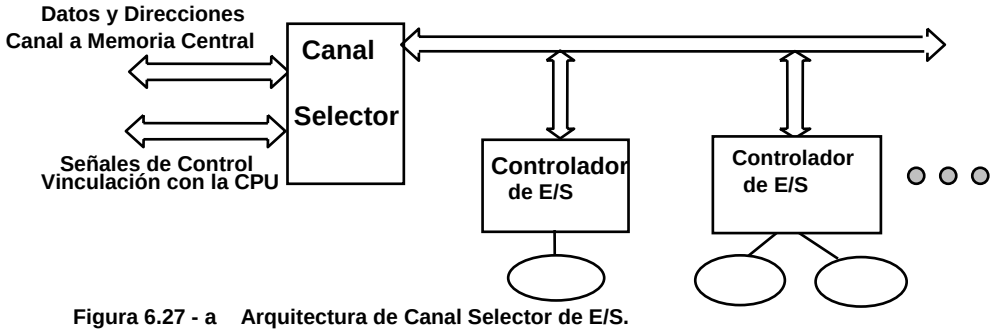


Figura 6.27 - a Arquitectura de Canal Selector de E/S.

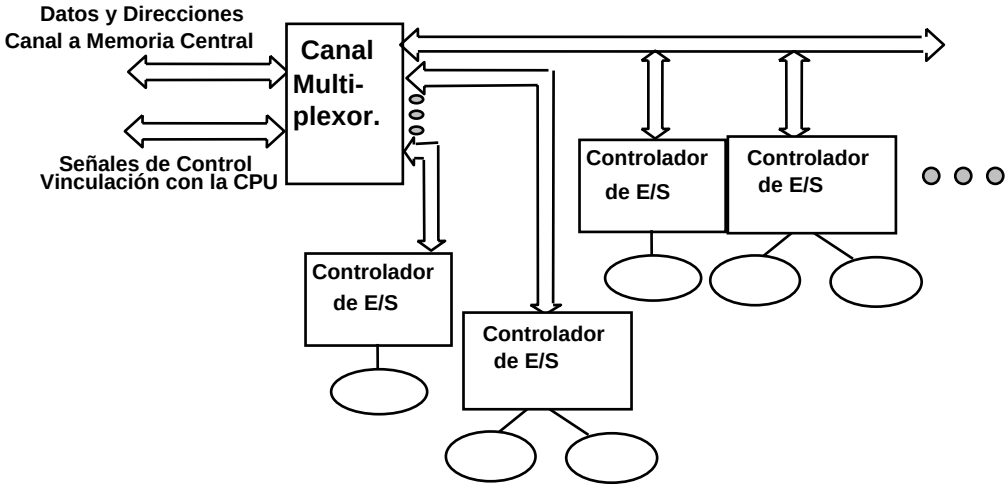


Figura 6.28 – b Arquitectura de Canal Multiplexor de E/S.

Arquitectura de Canal de E/S en el IBM® System/370

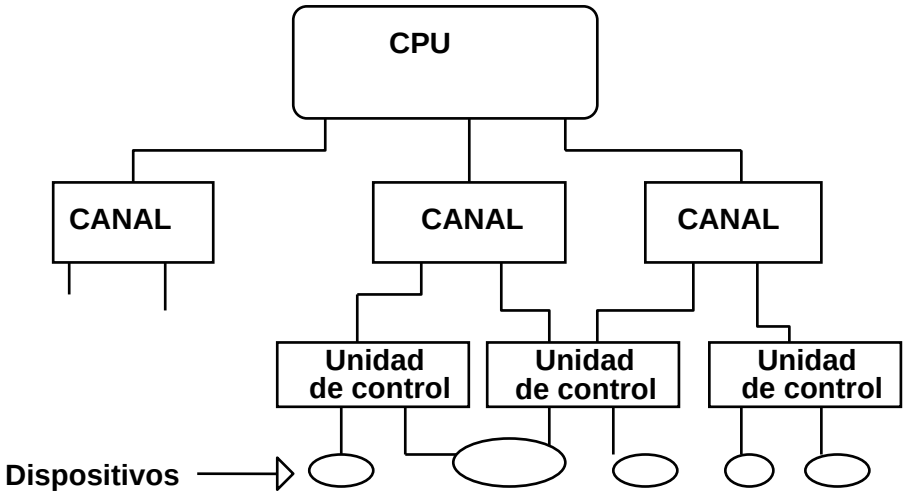


Figura 6.29 Arquitectura de canal de E/S en el System/370

Tradicionalmente, el uso de canales de E/S ha sido asociado con equipos mainframes o de gran escala. Esto ha sido así porque (1) los mainframes generalmente tienen una gran cantidad de almacenamiento secundario en discos y cintas y (2) algunos mainframes son utilizados intensivamente como sistemas multiusuarios de tiempo compartido y procesamiento de transacciones, soportando a más de 100 usuarios al mismo tiempo. Con el desarrollo de microprocesadores de bajo costo, el uso de canales de E/S se extiende ahora a los minicomputadores y también a los microcomputadores. Aún así, el canal de E/S completamente desarrollado se estudia mejor en el caso de un mainframe, y el ejemplo más conocido es el utilizado por IBM en sus sistemas de gran porte. Si bien ahora IBM dispone de una gran variedad de modelos de mainframes, la mayoría está basada en la arquitectura del System/370. Más aún, la "Biblia" o documentos de definición de los mainframes de IBM todavía hacen referencia a las arquitecturas como la del System/370. Es sobre esta arquitectura en la que basamos nuestro ejemplo.

Estructura

La descripción general dada anteriormente sobre una estructura del sistema usando canales de E/S se aplica a la arquitectura del IBM 370. La Figura 6.29 describe, en líneas generales, la estructura del subsistema para la familia IBM 370. El procesador controla uno o más canales. Un canal puede ser multiplexor de bytes o multiplexor de bloques. Cada canal controla uno o más controladores de E/S, llamados *unidades de control* (control unit). A una unidad de control típicamente se encarga de un conjunto de dispositivos similares o idénticos. Un ejemplo es un controlador de disco el cual controla varias unidades (drives) de disco.

Como se indica en la figura 6.29, es posible que una unidad de control se conecte a varios canales y que un dispositivo esté conectado a varias unidades de control. Esto provee más de una trayectoria física (path) entre el procesador y el dispositivo. Luego si una trayectoria está ocupada o está deshabilitada, una trayectoria alternativa puede encontrarse.

La arquitectura del IBM 370 utiliza un esquema de direccionamiento de E/S aislado para referenciar a los dispositivos. Las direcciones en el 370 son de 24 bits de longitud, y la Figura 6.30-a muestra el formato de la dirección de dispositivos. Las direcciones de dispositivos son de 16 bits de longitud; los 8 bits a la izquierda están en cero (0). Estos 8 bits son utilizados para designar el canal; luego con 8 bits se permite hasta 256 canales. Cuatro bits designan la unidad de control dentro del canal, y los cuatro bits restantes designan al dispositivo dentro de la unidad de control. Cuando están disponibles caminos alternativas, un dispositivo tendrá una dirección diferente para cada camino (path).

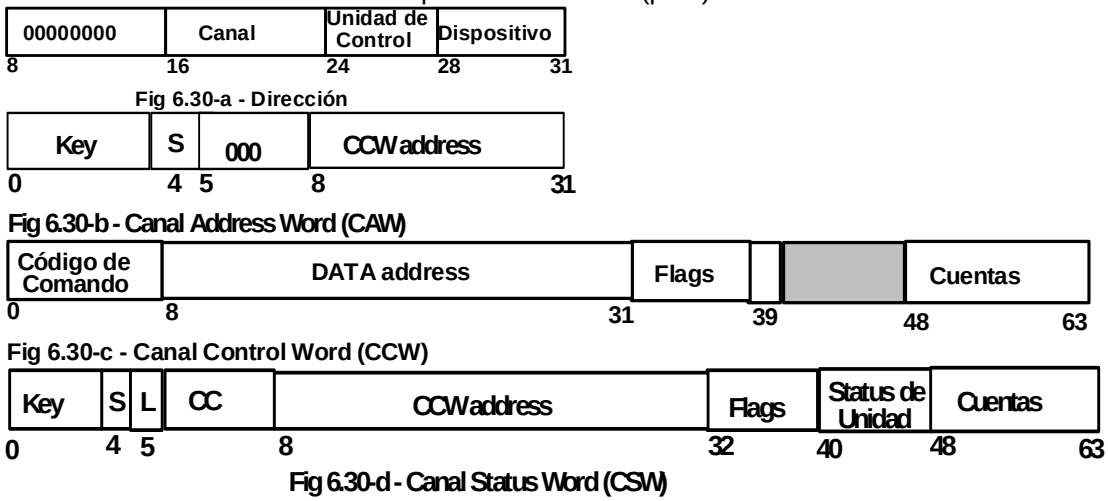


Figura 6.30- Formatos del Canal de E/S del System 370 de IBM®

Función:

La arquitectura de gran escala de IBM® define tres tipos de directivas de E/S :

* Instrucciones de E/S de CPU
* Comandos de canal
* Ordenes de Unidades de Control

Cada tipo es usado para controlar el procesamiento de E/S a un nivel diferente de hardware y provee un nivel diferente de detalle concerniente a una operación de E/S. Esto se explica mejor haciendo referencia a la Figura 6.31, la cual indica los siguientes pasos:

1. El procesador controla las operaciones de E/S con un pequeño y genérico conjunto de instrucciones de E/S emitidas a un canal de E/S.

2. Las instrucciones más detalladas al canal de E/S están especificadas en Palabras de Comando de Canal (Channel Command Words) (CCW) contenidas en memoria central. Las CCW son instrucciones o comandos que son obtenidos (fetched) y ejecutados por el procesador de E/S.
3. Basándose en los comandos especificados en las CCW, el canal emite órdenes de Unidad de Control a una de sus unidades de control. Durante el progreso de una operación de E/S, la información de control puede ser intercambiada desde y hacia entre el canal y la Unidad de Control.
4. Basándose en las órdenes de Unidad de Control que recibe provenientes del canal, la unidad de control emite señales de control a uno de sus dispositivos. Otra vez, las señales pueden ser intercambiadas en un sentido o el otro en el transcurso de la transferencia de datos. Estas señales de control representan un cuarto nivel de directivas de E/S. Sin embargo, esto toma lugar fuera de los límites del sistema 370, el cual se considera como una interfase de canal/Unidad-de-control, y en consecuencia no es parte de la arquitectura del 370.

5,6,7. Para una lectura (read) los datos son transferidos desde el dispositivo al canal, y luego del canal directamente a la memoria central. Una operación de escritura sigue la secuencia opuesta.

8. Siguiendo la completitud exitosa o no de una operación de E/S, el canal emite una interrupción a la CPU.

Examinemos ahora algunos detalles de ésta secuencia. Las instrucciones de E/S en las máquinas del tipo 370 son ejecutables por la CPU. La ejecución de una instrucción de E/S resulta de un comando del procesador que fue emitido y especificando la dirección de un canal, junto a las direcciones de su unidad de control y el dispositivo. Como se puede ver, el número de instrucciones de E/S es pequeño.

Una operación de E/S salida comienza cuando el procesador ejecuta una instrucción START I/O (SIO) o una instrucción START I/O FAST RELEASE (SIOF). El canal responde leyendo la Channel Address Word (CAW) almacenada en la posición 72 de la memoria central. Es responsabilidad del programa que emitió la SIO o la SIOF de almacenar en la posición 72 con la CAW deseada. La CAW contiene los siguientes campos (Figura 6.30-b):

- *Key*: una clave de acceso de 4 bits asociada con cada bloque de memoria de 2 o 4 Kbytes. La clave (key) en la CAW es utilizada por el canal cada vez que hace referencia a una posición de memoria central durante la operación de E/S.
- *S bit*: Cuando está en 1, indica que el procesador puede suspender y luego continuar con la operación de E/S.

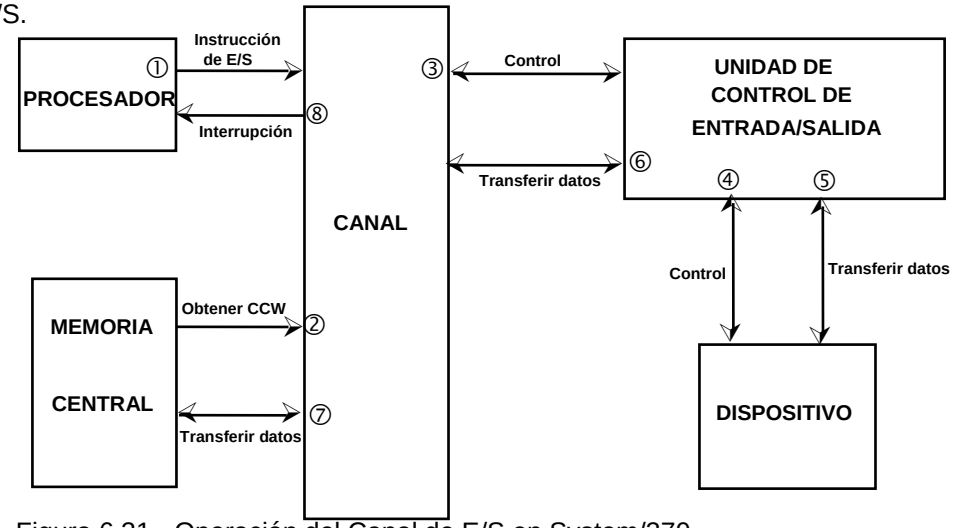


Figura 6.31 - Operación del Canal de E/S en System/370

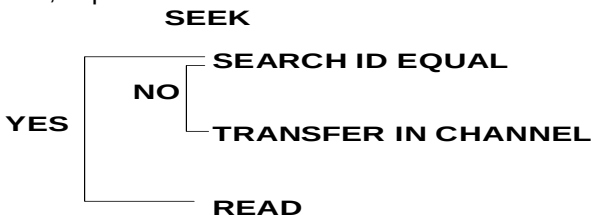
- *CCW address*: La ubicación del primer CCW a ser utilizada para ésta operación.

o El canal descodifica ésta información y procede a obtener (fetch) el CCW referenciado. El canal ejecuta ahora un programa de E/S consistente en uno o más CCWs. Otra vez, estos deben estar almacenados previamente en memoria. El CCW contiene los siguientes campos:

- *Command code*: Le dice al canal qué tipo de operación debe ejecutar. De nuevo, los tipos de comando son pocos y más bien generales. Los códigos de comandos incluyen bits modificadores que son específicos de cada dispositivo. Por ejemplo, los comandos de disco incluyen instrucciones SEEK e instrucciones para leer o escribir información de control así como los datos.
- *Data address*: Especifica la dirección en memoria de comienzo para la transferencia de datos (lectura o escritura)
- *Flags*: Especifican información adicional acerca de la operación a ser ejecutada.

- *Count*: Especifica el número de bytes a ser transferido en esta operación.

Bajo este esquema, un programa de E/S consiste en una o más CCWs. Múltiples CCWs son empleadas utilizando encadenamiento (chaining) y bifurcación. Las CCWs pueden ser encadenadas usando o bien encadenamiento de datos o encadenamiento de comando. En cualquier caso, el canal obtendrá, al completar un comando, la próxima CCW en la secuencia.



El comando TRANSFER IN CHANNEL se usa para permitir que las CCWs sean ejecutadas fuera de secuencia. Por ejemplo, un programa para leer un registro con un identificador ID específico de un disco podría tomar la siguiente forma:

- Los primeros dos comandos son modificaciones específicas al dispositivo de comandos generales CCWs. Primero, la operación SEEK posiciona la cabeza de lectura/escritura sobre la pista requerida. Luego, los registros son leídos hasta que coincide el ID. Esto causa que el programa salte al comando READ, el cual transfiere los datos.
- Luego de completar una operación, el canal almacena el estado de la operación en la Channel Status Word (CSW), en la posición 64 de la memoria, y emite una interrupción a la CPU. El procesador puede determinar el resultado de su instrucción leyendo la CSW, el cual contiene los siguientes campos:
  - *Key*: la clave originalmente especificada en la CAW para esta operación.
  - *S bit*: Indica si la operación fue suspendida o terminada.
  - *L bit*: Indica que esta pendiente un *logout*. Esto significa que alguna información de status específica a un dispositivo está disponible para la CPU. La instrucción de E/S solicitada no puede ser ejecutada hasta que el logout sea limpiado.
  - *Deferred Condition Code* (CC): Puede especificar una condición que se levanta durante una SIOF y que no fue informada inmediatamente porque el procesador fue liberada cuando el CAW fue obtenida (fetched).
  - *CCW Address*: La dirección en memoria central del byte que sigue a la última CCW que fue ejecutado.
  - *Unit Status*: Informa el status del dispositivo involucrado en la operación de E/S que causó el almacenamiento de la CSW.
  - *Channel Status*: Informa el estado del canal involucrado en la operación de E/S que causó el almacenamiento de la CSW.
  - *Count*: La cuenta residual, si hay alguna, relativa a la ejecución de la última CSW. La cuenta de la CCW menos la cuenta de la CSW iguala al número de bytes actualmente transferidos.

Arquitectura Extendida del System/370

En 1981, IBM introduce la Arquitectura Extendida (XA) del System/370 para sus sistemas de gran escala. La XA es diseñada para tomar provecho de los cambios en tecnología y los recursos del sistema disponibles desde de la introducción de la arquitectura 370. El cambio más visible es el direccionamiento de 31 bits en vez de 24. Sin embargo, se ve afectada la arquitectura virtualmente en todos sus aspectos, incluyendo el subsistema de E/S.

Varias tendencias en los sistemas de gran escala son relevantes:

- Velocidades de ejecución más rápidas.
- La utilización de microprocesadores en vez de lógica cableada en las unidades de control de E/S.
- La utilización de múltiples CPUs.

Las primeras dos tendencias están en conflicto. Con el uso de los microprocesadores, el tiempo requerido para que un dispositivo responda a una interrogación del procesador se aumentó. Luego la demoras en el procesador se incrementaron. Adicionalmente, el uso de múltiples CPUs requiere de la coordinación para compartir las unidades de E/S y los dispositivos. Uno de los objetivos primordiales de la XA es mejorar la perfomance global del sistema minimizando la intervención del procesador en el proceso de E/S.

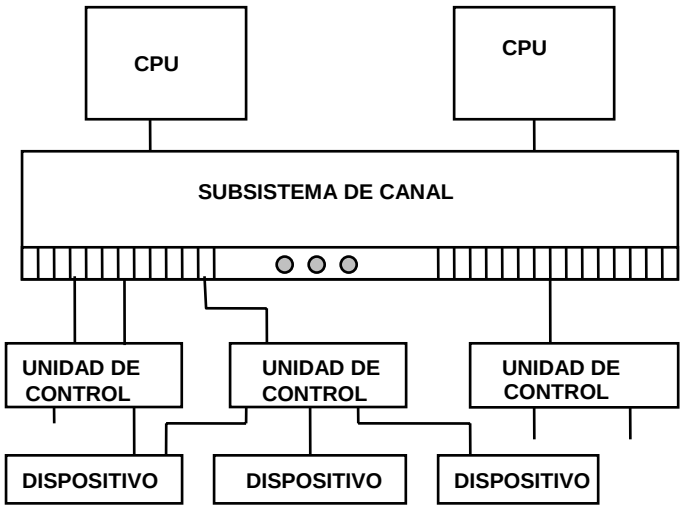


Figura 6.32 - Estructura E/S, System/370 XA

La clave para mejorar el rendimiento es la introducción de un subsistema de canales, como se muestra en la Figura 6.32. El subsistema de canales controla todos los dispositivos de E/S a través de *subcanales*. Desde el punto de vista del procesador, cada dispositivo tiene un único *número de subcanal lógico*. El subsistema de canales puede emplear una o más trayectorias (path) físicas para conectarse con el dispositivo. La existencia de diferentes trayectorias y de las unidades de control es ahora transparente a la CPU. Los detalles de la manipulación de la información de status e interrupciones de esta arquitectura de menor nivel es ocultada a la CPU.

La mejora en el rendimiento del sistema surge por la creciente potencia de procesamiento del subsistema de E/S. Esto permite el desplazamiento de funciones desde el procesador hacia el subsistema de E/S, reduciendo la complejidad del programa de CPU y la sobrecarga (overhead). Por ejemplo, el subsistema de canal de E/S verifica la disponibilidad de canales y unidades de control y el monitoreo de sus estados.

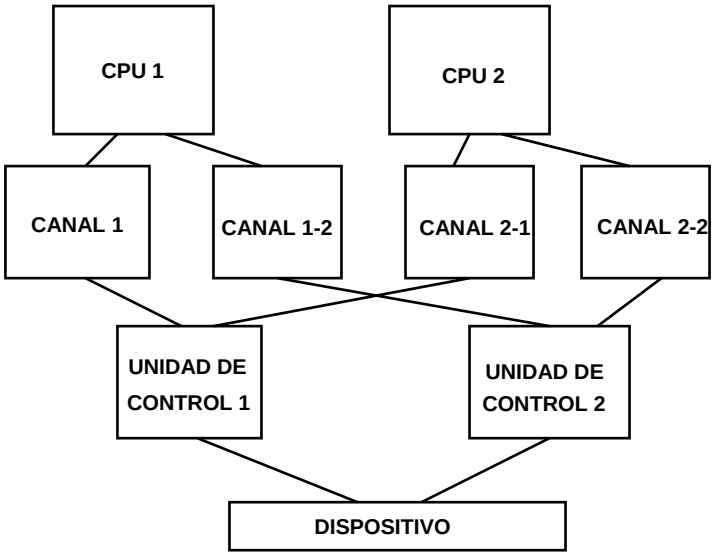


Figura 6.33 - Configuración de un Sistema de Multiprocesamiento

Como un ejemplo, considere el caso en la arquitectura 370 mostrada en la Figura 6.33. Con el System/370, si un sistema tiene múltiples CPUs, cada CPU tiene su propio conjunto de canales. A un programa en el procesador 1 le es posible solicitar un START I/O usando el Canal 1-1. Si se encuentra una condición de unidad ocupada (busy), puede intentar una solicitud en el Canal 1-2. Si este intento encuentra condición de canal ocupado, una interrupción puede ser emitida al procesador 2 para iniciar la solicitud desde ese procesador. El procesador podría requerir 2 intentos de START I/O para encontrar una trayectoria física disponible. En el System/370XA, una instrucción sería emitida al subsistema de canal, el cual manejaría los detalles.

Como lo implica la letra X, 370-XA es una extensión compatible con la arquitectura 370. Se utilizan los mismos formatos de la Figura 6.35.

La Interfase Externa

Tipos de Interfases: La interfase a un periférico desde un módulo de E/S debe ser diseñada de acuerdo a la naturaleza y operación del periférico. Una principal característica es si la interfase es serie o paralela (Figura 6.33). En una *interfase paralela*, hay múltiples líneas conectando el módulo de E/S con el periférico, y se transfieren múltiples bits simultáneamente, de la misma forma en que los múltiples bits de una palabra se transfieren en el bus de datos. En una *interfase serie*, hay una única línea para transmitir los datos, y los bits deben ser transmitidos una por vez. Las interfases paralelas se utiliza comúnmente para dispositivos de alta velocidad, como los discos y las cintas. La interfases serie son más comunes en las terminales e impresoras.

En cualquier caso, el módulo de E/S debe entrar en diálogo con el periférico. En términos generales, el diálogo para una operación de escritura es como sigue:

1. El módulo de E/S envía una señal solicitando permiso para transmitir datos.
2. El periférico acepta (acknowledge) la solicitud.
3. El módulo de E/S transfiere los datos (una palabra o un bloque dependiendo del periférico).
4. El periférico acepta (confirma) (acknowledge) la recepción de los datos.

Una operación de lectura tiene un procedimiento similar.

Un aspecto clave en la operación de un módulo de E/S es el buffer interno que puede almacenar los datos pasados entre el periférico y el resto del sistema. Este buffer permite al módulo de E/S compensar las diferencias en velocidad entre el bus del sistema y las líneas externas.

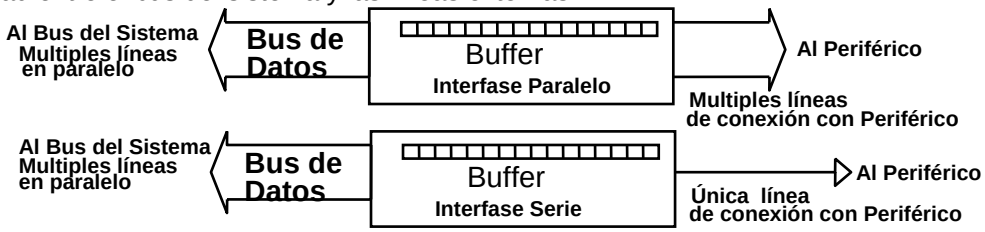


Figura 6.34 - E / S Paralela y Serie

Transferencia de Datos en Paralelo

El diálogo que toma lugar a través de la interfase incluye el intercambio de tanto información de control como de datos. Dos requerimientos fundamentales aparecen:

* Debe ser posible determinar cuando una señal empieza y cuando termina.
* Debe ser posible distinguir entre información de control y los datos.

Para la transferencia de datos en paralelo, ya hemos examinado soluciones a este requerimiento en éste módulo. La temporización puede ser sincrónica o asincrónica. En la temporización asincrónica, los eventos ocurren en secuencia, y un evento depende de la ocurrencia del evento previo. Las varias operaciones en el Unibus (bus único) exhiben este comportamiento. En la temporización sincrónica, lo eventos ocurren durante intervalos de tiempo específicos (slots), controlados por un reloj.

La distinción entre información de control y datos es simple en la transferencia paralela. Se dedican líneas diferentes a funciones diferentes.

Los ejemplos que hemos utilizado no son interfases externas pero sí buses internos del sistema. Sin embargo, los mismos principios se aplican a la interfase externa, como lo muestra el siguiente ejemplo.

***Interfase entre el Canal y la Unidad de Control en el System/370 de* IBM®**

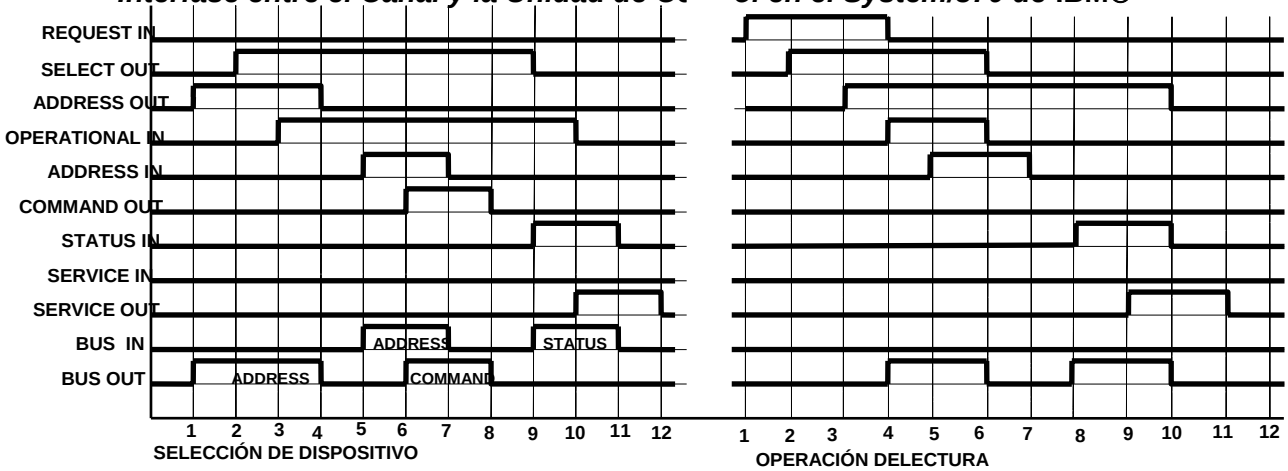


Figura 6.35 - Secuencia de Operaciones de Canal

Una de las interfases paralelo más ampliamente utilizada es la del System/370 entre los canales y las Unidades de Control. Como es típico en los computadores de gran escala, la arquitectura 370 determina que virtualmente todos los dispositivos se conectan a través de la Unidades de Control que a su vez se conectan al canal de E/S. El canal de E/S del System/370 no solamente es utilizado por el System/370, sino también para una variedad de fabricantes de periféricos compatibles, como los discos y las unidades de cinta. Más aún, esta interfase ha sido promulgada como un estándar del gobierno federal por la National Buerau of Standars (EEUU). En consecuencia, varios fabricantes proveen esta interfase de canal.

La interfase de E/S del System/370 es de propósito general. Provee un conjunto de 34 líneas de bus (extensible a 48). El canal y una unidad de control se conectan a este bus.

Los datos son transferidos entre el canal y la unidad de control un byte a la vez a través de las líneas BUS IN y BUS OUT. El canal controla el uso del bus seleccionado el dispositivo involucrado en la próxima operación y emitiendo comandos. Los bits de modificación se usan para adecuarlos a un tipo específico de unidad de control. Una unidad de control puede seleccionar un dispositivo específico por la dirección. Alternativamente, si se espera o desea tener actividad con más de una unidad de control, el canal puede emitir una señal de selección sin dirección (unaddressed). Cualquier unidad de control puede responder, y se emplea un esquema de prioridades de conexión encadenada (daisy-chain), usando las líneas SELECT IN y SELECT OUT. Estas líneas están enlazadas a través de todas las unidades de control de manera que la unidad que está físicamente más cerca del canal, tiene más alta prioridad, y así sucesivamente. La Figura 6.35 ilustra, de forma simplificada, la operación de la interfase. El ejemplo muestra la selección de un dispositivo seguido de la lectura de 1 byte. La selección de un dispositivo sigue los siguientes pasos. El canal coloca la dirección del dispositivo a ser seleccionado en las líneas BUS OUT y levanta la línea ADDRESS OUT para alertar a las unidades de control. El canal levanta después la línea SELECT OUT para esperar una respuesta. Si la unidad seleccionada está disponible, ésta señaliza su disponibilidad levantando la línea OPERATIONAL IN. El canal acepta bajando la línea ADDRESS OUT. La unidad de control coloca luego su dirección en el bus. Cuando esta coordinación se completa, el canal emite el comando de lectura y la unidad de control responde con su status corriente. El canal acepta con una señal SERVICE OUT.

Después de la selección inicial, la unidad de control descodifica el comando que ha recibido y lo envía al dispositivo para la ejecución. Mientras, el canal puede hacer otras cosas. Cuando la operación en el dispositivo se completa, éste solicita el uso del bus vía un REQUEST IN. Cuando recibe el permiso, vía un SELECT OUT, la unidad de control coloca la dirección del dispositivo en el bus. El canal acepta con COMMAND OUT, solicitando que el comando previo sea ejecutado. La transferencia de datos por fin toma lugar.

Secuencias más complejas son posibles. Una unidad de control puede enviar o recibir un bloque a la vez en vez de un byte a la vez, y tanto el multiplexado de byte o de bloque son permitidos.

Transferencia De Datos En Serie

En la transmisión serie, un único camino se utiliza para manejar de toda la información de control y de datos entre el módulo de E/S y el dispositivo. La transmisión serie se usa comúnmente para la conexión a una terminal y para las líneas de comunicaciones de datos. Con sólo una línea disponible para la transferencia de datos, se hace más dificultosa en la transmisión de datos proveer un esquema de temporización. Dado que una secuencia de bits serán transferidos en la línea, un requerimiento fundamental es que el receptor sepa el instante en que comienza y la duración de cada bit que recibe.

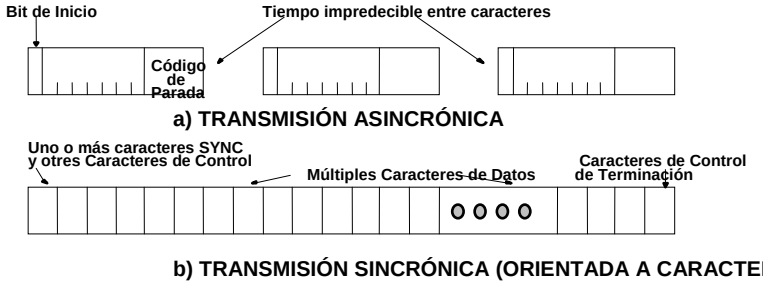


Figura 6.36- Transmisión asincrónica y sincrónica

El primer y más sencillo para cumplir este requerimiento es la ***transmisión asincrónica***. En este esquema, los datos se transmiten un caracter (de 5 a 8 bits) a la vez. Cada caracter es precedido por un código de inicio y seguido por un código de parada (Figura 6.36-a). El **código de inicio** (*code start*) tiene la codificación de un 0 y una duración de 1 tiempo de bit; en otras palabras, el código de inicio es 1 con un valor de 0. El **código de parada** (*stop code*) tiene un valor de 1 y una duración mínima, dependiendo del sistema, de 1 o 2 tiempos de bit. Cuando no hay datos que enviar, el transmisor envía un código de parada

en forma continua. El receptor identifica el comienzo de un nuevo caracter por la transición de 1 a 0. El receptor debe tener una idea bastante precisa de la duración de cada bit para poder recuperar todos los bits de un caracter. Sin embargo, una pequeña cantidad de desvío respecto de esta duración (por ejemplo, 1 %) no tiene importancia dado que el receptor se re-sincroniza con cada código de parada. Este medio de comunicación es simple y barato, pero requiere una sobrecarga (overhead) de 2 ó 3 bits por caracter. Esta técnica se la conoce normalmente como *asincrónica* porque los caracteres son enviados independientemente uno de otro, y el transmisor y el receptor no están sincronizados por un reloj compartido. Luego los caracteres pueden ser enviados a una velocidad (de caracteres) no uniforme.

Un medio más eficiente de comunicación es la ***transmisión sincrónica***. En este modo, bloques de caracteres o bits son enviados sin códigos de inicio o parada, y el tiempo exacto de partida o llegada de cada bit es predecible. Para prevenir un desvío del tiempo entre el transmisor y el receptor, sus relojes deben de alguna forma estar sincronizados. Una posibilidad es usar una línea de reloj separada entre el transmisor y el receptor. De otra forma, la información de reloj debe estar incluida en la señal en sí misma, por alguna forma de codificación.

Con la transmisión sincrónica, hay otro nivel de sincronización requerido, para permitir al receptor determinar el comienzo y el final de cada bloque de datos. Para obtener esto, cada bloque empieza con un patrón de bits ***preámbulo*** y termina con un patrón de bits ***postámbulo*****.** Los datos junto al preámbulo y al postámbulo, recibe el nombre de **cuadro (*frame*).** La naturaleza del preámbulo y del postámbulo depende de si el bloque de datos es orientado a caracter o a bit.

Con los esquemas *orientados a caracter*, cada bloque es precedido por un o más "caracteres de sincronización" (Figura 6.36-b). El caracter de sincronización, usualmente llamado SYNC, se elige de tal forma que el patrón de bits es significativamente diferente de cualquier otro caracter regular a ser transmitido. El postámbulo es otro caracter único y el receptor acepta datos hasta que el caracter de postámbulo se recibe. El receptor puede luego mirar por el próximo patrón de SYNC.

Los esquemas orientados a caracter, como el BCD de IBM, están gradualmente siendo reemplazados por *esquemas orientados a bit* más eficientes y flexibles, los cuales tratan a los bloques de datos como un flujo de bits en de un flujo de caracteres. Presentamos un esquema orientado a bit en lo que sigue.

Para la transmisión serie, la información de control y los datos no pueden ser separados físicamente en líneas diferentes. En cambio, se utilizan convenciones de formato. Esto se ilustra en el ejemplo que sigue.

High Level Data Link Control (HDLC)

Una de las técnicas más comunes de transmisión serie es HDLC. Desarrollada por la ISO (International Organization for Standarization), HDLC está disponible de la mayoría de los fabricantes tanto como intercambio de datos de computador a computador como de terminal a computador.

Se definen tres modos de operación: *modo de respuesta normal* (NRM), *modo de respuesta asincrónica* (ARM), y *modo asincrónico balanceado* (ABM). Tanto NRM como ARM pueden ser utilizados en configuraciones punto-a-punto o configuraciones multipunto. Para cada una hay una *estación primaria* y una o más *estaciones secundarias*. La estación primaria tiene la responsabilidad de inicializar el enlace, controlar el flujo de datos desde y hacia las estaciones secundarias, de la recuperación contra errores, y de desconectar lógicamente a las estaciones secundarias. En el modo NRM, una estación secundaria puede transmitir solo en respuesta a una interrogación o escrutinio (poll) de la estación primaria; en el modo ARM, la estación secundaria puede iniciar una transmisión sin una interrogación (poll) previa. NRM es ideal para una línea multipunto consistente en un computador central (host) (estación primaria) y un número de terminales (estaciones secundarias). ARM puede necesitarse en ciertas clases de configuraciones en lazo (loop).

Estructura del cuadro (frame)

8 bits	8	8	0	16	8
Flag	Address	Control	Datos	CRC	Flag

Estructura de los Campos de Control

	1	2	3 4	5	6 7 8
Información	0	Seq		P/F	Next
Supervisión	1	0	Type	P/F	Next
No numerados	1	1	Type	P/F	Modifier

Figura 6.37- La estructura de cuadro del HDLC

El modo ABM sólo se utiliza en configuraciones punto-a-punto, y cada estación asume el rol de primaria y secundaria. El modo ABM es más eficiente para las líneas punto-a-punto ya que no aparece el overhead por la interrogación y ambas estaciones pueden iniciar transmisiones.

Los datos se transmiten en cuadros que consisten de 6 campos (Figura 6.37).

- *FLAG*: Usado para la sincronización; estos campos indican el comienzo y final de un cuadro.
- *ADDRESS*: Este campo identifica a la estación secundaria de esta transmisión.
- *CONTROL*: Este campo indica la función y propósito de este cuadro.
- *DATA*: Este campo contiene los datos a ser transmitidos.
- *CRC*: Este es el campo de verificación de la secuencia. Utiliza una *verificación por redundancia cíclica* de 16 bits (CRC). El campo CRC es una función del contenido de los campos de *address*, *control* y *data*. Es generado por en transmisor y luego por el receptor. Si el valor de la función (resultado) calculada por el receptor difiere del valor presente en el campo CRC, se asume que un error en la transmisión ha ocurrido.

Se utilizan tres tipos de cuadros, cada un con un formato del campo de control diferente. Los cuadros de información portan datos. Los cuadros de supervisión proveen las funciones de control de enlace básicas, y los cuadros no numerados proveen funciones adicionales de control de enlace.

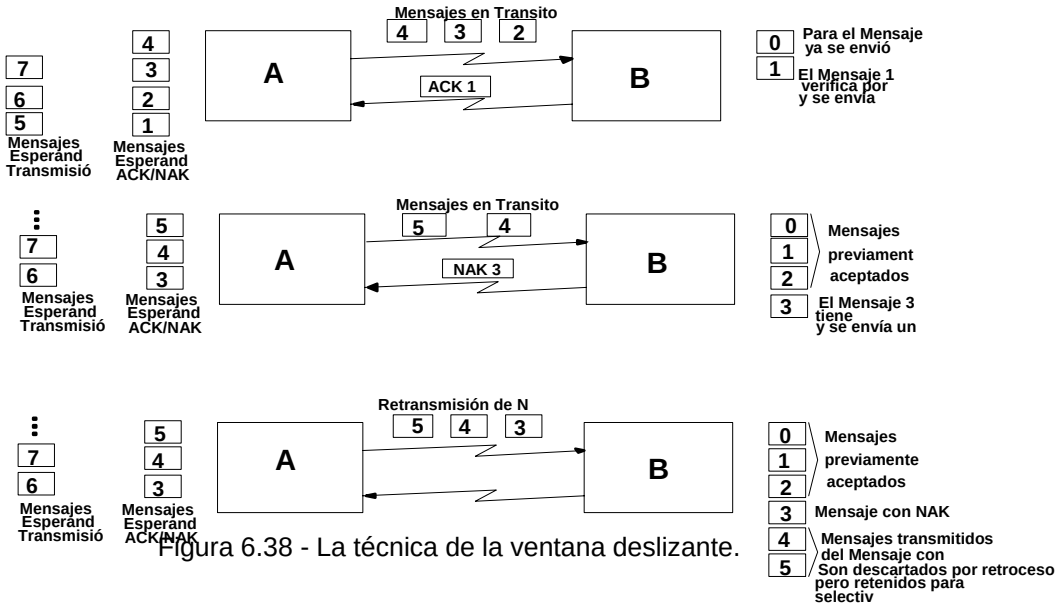


Figura 6.38 - La técnica de la ventana deslizante.

El bit P/F (poll/final) se utiliza por la estación primaria para solicitar una respuesta. Más de un cuadro puede ser enviado en respuesta con el bit P/F en 1 para indicar el último cuadro. El bit P/F puede ser utilizado en los cuadro de supervisor o no numerados para forzar una respuesta.

Los campos SEQ y NEXT en los cuadros de información proveen una técnica eficiente para el control de flujo y el control de errores. Una estación numera los cuadros que envía secuencialmente, usando el campo SEQ. Cuando una estación recibe un cuadro de información válido, acepta (acknowledge) ese cuadro con su propio cuadro de información almacenando en el campo NEXT el número de cuadro siguiente que espera recibir. Esto se conoce como *piggybacked acknowledgement*, dado que la aceptación (acknowledge) viaja de vuelta en un cuadro de información. Las aceptaciones pueden enviarse también en cuadro de supervisión. Este esquema logra tres funciones importantes.

- *Control de Flujo*: Una vez que una estación ha enviado 7 cuadros, no puede enviar más hasta que el primer cuadro sea contestado (y aceptado).
- *Control de Error*: Si un cuadro es recibido con error, una estación puede enviar una NAK (contestación negativa) vía un cuadro de supervisión para especificar cual cuadro fue recibido con errores. Esto se hace de dos formas. En el protocolo *go back n*, la estación transmisora retransmite el cuadro negado (NAK´ed) y todos los cuadros subsecuentes que ya habían sido enviados. En la *técnica de selección de repetición*, la estación sólo retransmite el cuadro con el error.
- *Pipelining*: Más de un cuadro puede estar en tránsito a la vez; esto permite una utilización más eficiente de enlaces con alta demora en la propagación, como lo son los enlaces satelitales.

La técnica SEQ/NEXT se conoce como *protocolo de ventana deslizante* porque la estación transmisora mantiene una ventana de mensajes a ser enviados que gradualmente avanza durante la transmisión y las contestaciones (acknowledges) positivas.

Hay cuatro tipos de cuadros de supervisión:

* *Receive Ready (RR)*: Utilizado para aceptar la correcta recepción de cuadros hasta el valor NEXT-1. Además, se utiliza RR como un comando de interrogación instruyendo a la estación secundaria a comenzar la transmisión con el número de secuencia NEXT.
* *Receive Not Ready (RNR)*: usado para indicar una condición temporaria de ocupado. El campo NEXT se usa para una aceptación redundante.
* *Reject (REJ)*: Usado para indicar un error en el cuadro NEXT y para requerir la retransmisión de ese y todos los cuadros subsecuentes.
* *Selective Reject (SREJ)*: Usado para requerir la retransmisión de un único cuadro.

Los cuadros no numerados no tiene números de secuencia y se utilizan para una variedad de propósitos especiales, como ser inicializar a una estación, establecer el modo, desconectar a un estación, y rechazar (reject) un comando.

6.6. Bibliografía recomendada para este Módulo

1. Operating Systems. (Fourth Edition), Stallings Willams; Prentice Hall, Englewood Cliff, NJ. 2000, 779 pages.
2. Operating Systems Concepts (Fifth Edition), Silberschatz, A. and Galvin P. B; Addison Wesley 1998, 850 Pages.
3. Operating Systems Concepts and Design (Second Edition), Milenkovic, Millan; Mc Graw Hill 1992
4. Modern Operating Systems, Tanenbaum, Andrew S.; Prentice - Hall International 1992, 720 pág.
5. Operating Systems, Design and Implementation, Tanenbaum, Andrew S.; Prentice - Hall International, 1987, 800 pág.

Glosario De Términos En Idioma Inglés

I/O Scheduler	Input	Output	Mainframe
Host	Ports	Port in	Port out
File System	Binding	Spooler	Buffered
Buffering	Read Sector	Write Sector	Sleep
Fetch	Seek	Scan	Record ID
Busy	Ready	Not ready	Burst
Paper Jam	To Interface	Open	Close
Loop	Status	Status word	Handshaking
Wait for I/O	Iddle	Preemptive	Direct Memory Access
Search time	Seek time	Channel time	Bootstrapping
Bootstrap Program	Queueing time	Data transfer time	Latency time
Off - Line	Handler	Path	Get
Put	Catode Ray Tube	Polling	Acknowledgement
Enquiry	End of Transmmition	End of Text	Start of heading
Negative Acknowledgement	Syncronous	File separator	Group Separator
Record separator	United Separator	Gaps	Backspace
Horozontal tab	Line feed	carriage return	null
Bell	Shift out	Shift in	Delete
Space	Data link	Device Control	Device Driver
Drive	Driver	Cancel	Overlap
Home address	Interleave	Layered	Header
Data	Boot sector	Kernel	File Allocation Table
Swapping	Scan	Caching	Look-up
Track	Thrashing	First Come First Served	Shortest seek time First
Shortest Job First	Disk interleaving	Disk Stripping	Mirroring
Block	Shadowing	Block interleaved Parity	Arrays
Clock	Time of day clock	Battery backup	Squere wawe
Watchdog timers	Interrupt driven	Test	Daisy chain
polling	Data Ready	Cold Start	I/O Page
I/O Address	Mode Raw	Mode Cooked	Context Switch
Time Out	Daemon	Echoing	Control Unit
Key	Cannel Command worth	Status I/O Fast Release	Command code
Counts	Flags	Search ID equal	Transfer IN channel
Logout	Login	Unit Status	Channel Status
Slots	Unaddressed	Start code	Stop Code
Frame	Piggybacked	Next	Receive ready
Rejrct	American Standard Code for information interchange	Logical Input Output Control System	Phisical Input Output Control System

Glosario De Términos En Castellano

Sistema de gestión de E/S	Entrada
Salida	Dispositivos de E/S
Conexión de periféricos al Bus del Sistema	Funciones del Módulo de E/S
Sistema de Gestión de E/S	Sistema de Gestión de Archivos
Control y Temporización	Comunicación con el dispositivo
Detección de errores	Estructura del módulo de E/S
Canal	Subcanal
Procesador de E/S	Controlador de Dispositivos
Señales de diálogo	Operación asincrónica
Operación sincrónica	Diferencia de velocidades
Buffering	Tiempo de respuesta de E/S
Dispositivos orientados a Bloques	Dispositivos Orientados a caracteres
Dispositivos legibles para el ser humano	Dispositivos legibles para la máquina
Dispositivos de comunicaciones	Unidad de control de Dispositivos
Controlador	Registro de estados
Registro de órdenes	Registro de Buffers
Estado de Dispositivos	Dispositivos externos
Señales de estados	Lógica de control
Transductor	Disco
Video terminal	Espacio de swapping
Algoritmos de planificación del brazo del Disco	Relojes
Técnicas de E/S	E/S programada
E/S por interrupciones	E/S por DMA
Instrucciones de E/S	E/S aislada
E/S por correspondencia en Memoria	Robo de ciclo
Niveles de servicios de E/S	Software de E/S
Metas del Software de E/S	Independencia de los dispositivos
Device driver	Tipos de Device driver
Pasos y controles de una operación de E/S	

Acrónimos Usados En Este Módulo

E/S	Entrada / Salida	Px	Proceso x
DMA	Direct Memory Access	FF	First Fit
PCB	Process Contol Block	BF	Best Fit
I/O	Input/ Output	WF	Worst Fit
RAM	Random Access Memory	ID	Identifier
ROM	Read Only Memory	IDE	Integrated Drive Electronics
CPU	Central Processing Unit	SCSI	Small Computer System Interface
M.C.	Memoria central	LAN	Local Area Network
S.O.	Sistema Operativo	MODEM	Modulation Demodulation
R/W/	Read / Write/Execute	EISA	Extended Industrial Standard Architecture
PIT	Programmable Interval Timer	ISA	Industrial Standard Architecture
UC de E/S	Unidad de control de E/S	PCI	Peripherical Component Interconect
IOP	Input Output Processor	UC	Unidad de Control
VDT	Video Display Terminal	UART	Universal Asyncrinous Receiver Transmiter
CRT o TRC	Catode Ray Tube	r.p.m.	Revoluciones por minuto
VTOC	Volumen Table Of Contend	WORM	Write Once Read Meny
CD-ROM	Compact Disk ROM	Sync	Syncronismous
Cyl	Cylinder	HD	Head
Sec	Sector	ECC	Error Code Corection
CRC	Chec Redundance Code	FAT	File Allocation Table
FCFS	First come First served	SSTF	Shortest Seek Time first
SJN	Shortest job next	RAID	Redundant Array of Inexpensive Disk
PSW	Program Status Word	P.C.	Program Counter

INTR	Interrupt Request	INTA	Interrupt Acknowledgement
POICS	Phisical Input Output Control System	LIOCS	Logical Input Output Control System
PAI	Pedido de Atención de una Interrupción	PES	Pedido de Entrada /Salida
EOF	End of File	BCD code	Binary Codified Decimal code
BCD	Bloque de Control del Driver	BCU	Bloque de control de Unidad (Dispositivo)
CCW	Channel Command Word	CAW	Channel Address word
CSW	Channel Status Word	CC	Condition code
SIO	Start I/O	SIOF	Start I/O Fast Release
XA	Extended Architecture	HDL	High Level Data Link Control
NRM	Normal Response Mode	ABM	Asyncrinic Balanced Mode
ARM	Asyncronic Response Mode		
ASCII	American Standard Code for Information Interchange		

Anexo 6.a: Cuestiones de diseño en los Sistemas Operativos para E/S

Objetivos

- **Eficiencia:** es importante porque las operaciones de E/S constituyen un cuello de botella del sistema. Los dispositivos de E/S son extremadamente lentos comparados con la memoria central y el procesador, a pesar de la multiprogramación, el mismo swapping es una operación de E/S.
- **Generalidad:** por simplicidad y exención de errores, es deseable manejar todos los dispositivos en forma uniforme, en cuanto a la forma en la que los procesos ven los dispositivos de E/S y la forma en la que el sistema operativo los administra y sus operaciones. La diversidad complica esto, aunque un enfoque jerárquico y modular disimula los detalles de la E/S con dispositivos en rutinas de bajo nivel (lectura, escritura, apertura, cierre, bloqueo y desbloqueo). Como ejemplo el UNÍX, ve a todos los dispositivos como archivos.

Estructura lógica de la función de E/S

En el caso de un dispositivo periférico local que se comunica de una manera sencilla, como un flujo de bytes o registros. los niveles involucrados entre el proceso de usuario usuario y el hardware son:

- ***E/S lógica*:** el módulo de E/S lógica se encarga del dispositivo como un recurso lógico y desconoce los detalles del dispositivo. Sólo administra funciones generales de E/S en nombre del proceso, permitiéndoles utilizar los dispositivos a través de un identificador de dispositivo y un comando como abrir, cerrar , leer y escribir.
- ***E/S con dispositivos*:** las operaciones requeridas y datos son convertidos en secuencias de instrucciones de E/S, comandos al canal y órdenes al controlador. Aquí es donde se utilizan las técnicas de Almacenamiento intermedio.
- ***Planificación y Control*:** La planificación y encolado de las operaciones de E/S ocurren en este nivel, y el control de las operaciones. También las interrupciones son manejadas en este nivel y el estado de E/S es recolectado e informado. Éste es el nivel del software que interacciona con el módulo de E/S, y por lo tanto con el hardware.

En el caso de un dispositivo destinado a las comunicaciones, la estructura de E/S intercambia al módulo de E/S lógica por la arquitectura de comunicaciones (la que puede consistir a su vez en un número de capas).

En el caso de E/S en un dispositivo de almacenamiento secundario que soporta un sistema de archivos, debemos incluir tres niveles más:

- ***administración de directorios*:** se traducen los nombres de archivos simbólicos a identificadores que referencian al archivo directa o indirectamente a través de un descriptor de archivo o una índice de una tabla. Este nivel es responsable de las operaciones que afectan a los directorios de archivos.
- ***Sistema de archivos*:** Es responsable de la estructura lógica de los archivos, permisos de acceso y operaciones del usuario.
- ***Organización física*:** las referencias lógicas a archivos y registros deben ser traducidas a direcciones físicas del dispositivo de almacenamiento secundario, teniendo en cuenta la estructura en pistas y sectores del archivo. La asignación de espacio y buffers de almacenamiento son realizadas en este nivel.

Como resumen podemos afirmar que la aplicación específica de la filosofía de E/S conduce a una clase de organización que depende del tipo de dispositivo y de la aplicación.

Se busca eficiencia, dado que los procesos de E/S son lentísimos (mseg.) frente a los tiempos del procesador (nseg.).

- Se necesitan interfases lógicas homogéneas para que el S.O. los vea a todos los dispositivos periféricos lo mas abstracto posible con un mínimo de operaciones (llamadas al sistema) en el alto nivel.
- La adaptación del software independiente (device Driver) del dispositivo al S.O. debe contemplar el tratamiento de interrupciones y errores, además de las interfases lógicas, tablas, inicialización, etc.
- Las operaciones deben ser multiprogramadas, de forma que el proceso libere el procesador cuando realiza una operación de E/S (se Bloquea) y otro proceso pueda recibir CPU.
- Toda la comunicación de datos y mensajes deben vincularse lógicamente y físicamente mediante tablas y punteros.
- La transferencia de datos deben ser sincronizadas por las diferentes velocidades de los buses.

- Se deben planificar y controlar todas las operaciones que realizan entre el proceso y los dispositivos desde el aspecto organizacional, administrativo y operacional.

Estructuras lógicas más importantes:

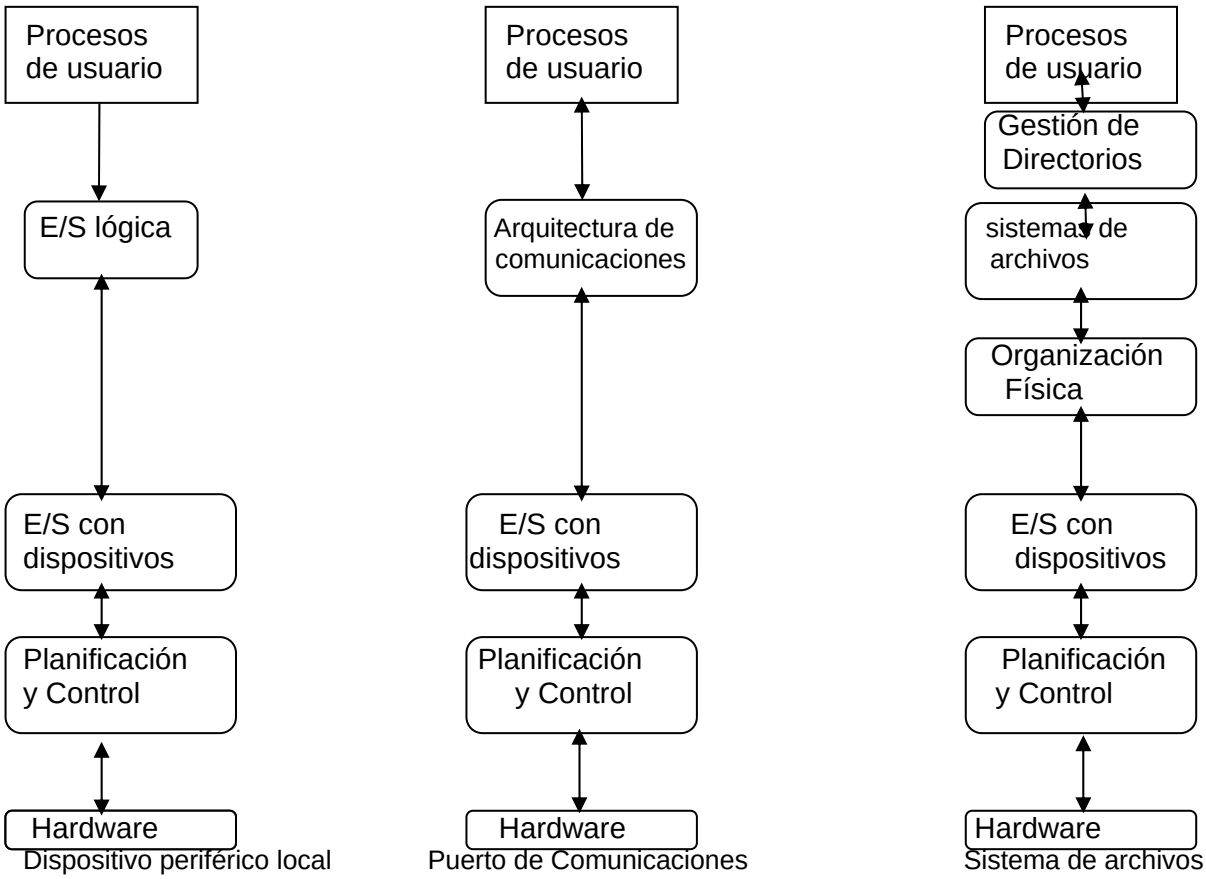


Fig 6.a.1 Estructuras lógicas de Entrada/Salida

La mayoría de los SO enfocan la E/S, más o menos, de la siguiente manera:

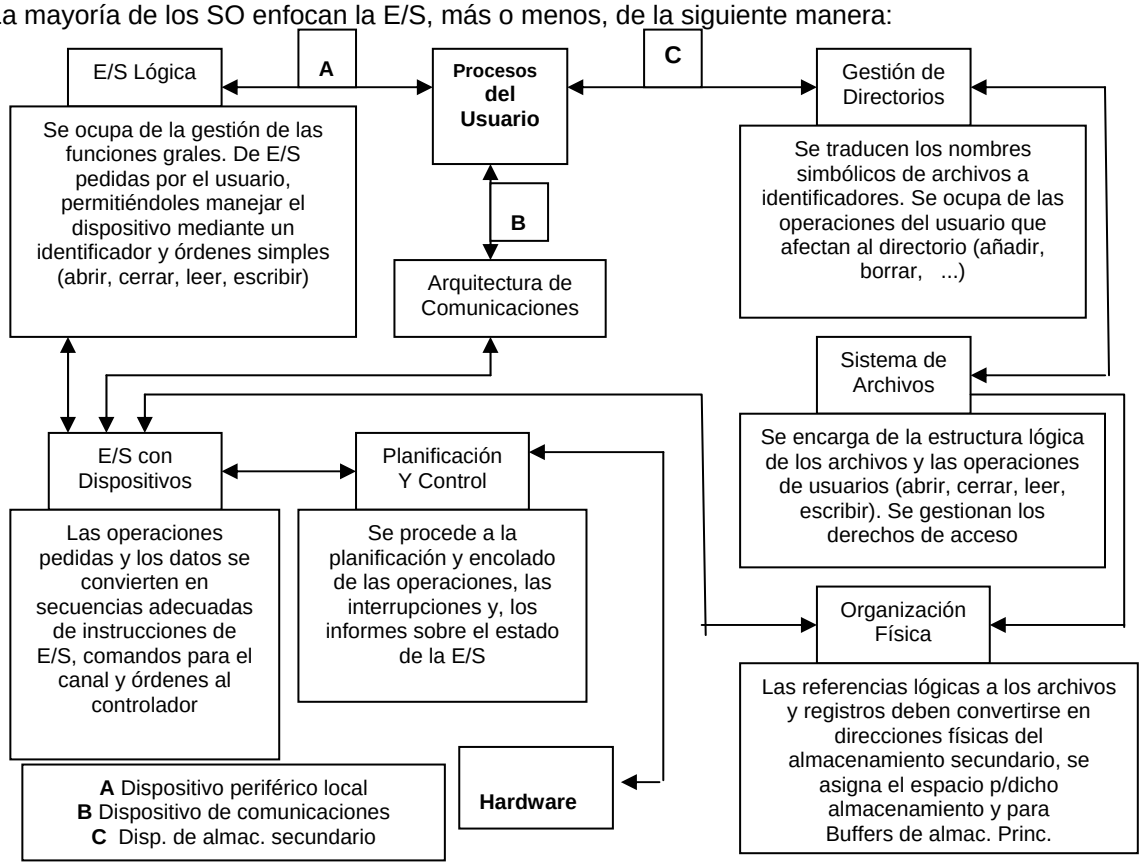


Fig. 6.a 2 Interacción de los módulos de E/S

Anexo 6.b: Ejemplo de E/S:

a) Entrada y Salida en Windows NT

Windows NT soporta trabajo en red con varios protocolos de comunicaciones. Lo más importante es que las facilidades de red están integradas en el S.O., lo que lo distingue de DOS y de la mayoría de las versiones de UNIX, en los que las interfases con la red eran un añadido al SO, y frecuentemente no se adaptaban del todo bien al mismo.

Vamos a describir brevemente qué ocurre en cada uno de los niveles de implementación del modelo OSI:

- En el nivel 0 aparece un dispositivo que es la tarjeta de interfase a la red (Network Interfase Card, NIC). El NIC conecta el bus interno de la máquina con la red, sirviendo de interfase entre el nivel 0 y el 1 (nivel físico). Es contemplado por el S.O. como un periférico más, controlado a través de su driver correspondiente.
- En el nivel 2 (nivel de enlace de datos) aparece un software llamado NDIS (Network Device Interface Specification), que es una interfase entre los drivers de dispositivo del NIC y los protocolos de transporte.
- En los niveles 3 (nivel de red) y 4 (nivel de transporte) Windows NT sitúa el software de los protocolos de transporte. Soporta TCP/IP, NBF, NWLink, DLC y AppleTalk.
- En el nivel 5 (de sesión) aparecen dos interfaces con los protocolos de transporte, que son los Windows Sockets (WinSock) y la NetBIOS.

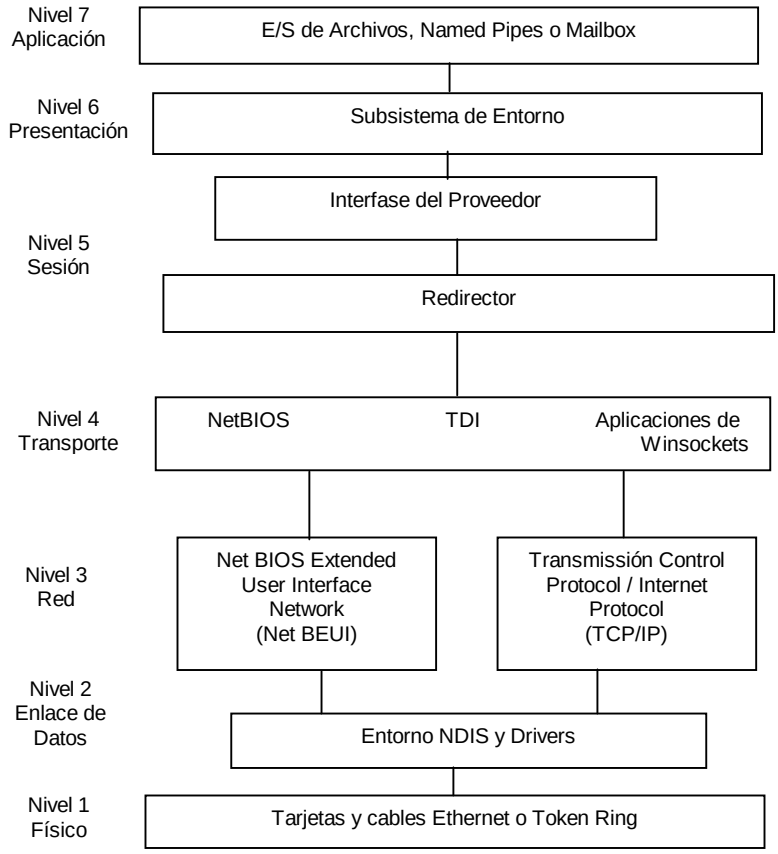


Fig. 6.b.1. Niveles de interacción de E/S en Windows NT

Un socket es un mecanismo para establecer una conexión entre dos máquinas. Actúa como una especie de tubería bidireccional para los datos. Fueron introducidos por primera vez por el UNIX de Berkeley, y Windows NT incorpora una versión especial llamada WinSock. Se utilizan cuando se quiere una comunicación a través del protocolo TCP/IP, IPX/SPX.

La interfase NetBIOS es usada por aquellas aplicaciones que deseen usar protocolos que se adapten a NetBIOS (como el NBF). Establece conexiones entre distintas máquinas de la red y se encarga de que la transmisión sea fiable una vez establecida la conexión.

En la misma capa de sesión están dos subsistemas integrales gestionados por el administrador de la E/S, denominados el redireccionador y el servidor. El redireccionador es el responsable de enviar peticiones de E/S a lo largo de la red, cuando el fichero o dispositivo solicitados son remotos. Al servidor le llegan peticiones desde los redireccionadores clientes, y las gestiona de modo que alcancen su destino.

Tanto servidores como redireccionadores son implementados como drivers del sistema de ficheros. Así, cuando un proceso quiere realizar una E/S, usará las mismas llamadas al sistema para acceso local o remoto, con lo que no necesita conocer la ubicación del recurso (fichero o dispositivo). Pueden existir múltiples parejas de redireccionadores-servidores ejecutándose concurrentemente.

En el nivel 6 (capa de presentación) define como se presenta la red a si misma ante la maquina y sus programas y/o aplicaciones.

En el nivel 7 (capa de aplicación) existe un proceso llamado suministrador por cada redireccionador de la capa 5. Cuando una aplicación hace una llamada de E/S, un software llamado enrutador de suministradores (multiple provider router) determina el suministrador adecuado, y le envía la petición. El suministrador, a su vez, se la pasará al correspondiente redireccionador. Por ejemplo, el gestor de ficheros (del administrador de E/S) es una aplicación que usa los servicios de los suministradores.

El último tema sobre E/S que vamos a tratar es la gestión de entradas del usuario (mediante teclado o ratón). Cuando el sistema se arranca y se crea el proceso subsistema Win32, este proceso crea a su vez un subproceso llamado subproceso de entrada inicial (Raw Input Thread, RIT), que por lo general está inactivo. Cada vez que un usuario pulsa una tecla, o mueve o pulsa el ratón, el driver de dispositivo correspondiente añade un suceso hardware a la cola de mensajes del RIT. Entonces, el RIT despierta, examina el mensaje, determina qué tipo de suceso es (WM_KEY*, WM_? BUTTON*, WM_MOUSEMOVE) y a qué subproceso va dirigido, y lo envía a la cola de mensajes de dicho subproceso. Para determinar el subproceso destino, el RIT examina sobre qué ventana se encontraba el ratón cuando se produjo el suceso, y determina qué subproceso es el propietario de ese objeto ventana. Si es una secuencia de teclas, el RIT busca qué subproceso está actualmente en primer plano, y a él le mandará el mensaje.

No obstante, el RIT también monitoriza qué tipo de entrada es, para que de esta manera se pueda cambiar de contexto (Alt-Tab), o llamar al proceso Administrador de Tareas (si Ctrl-Esc) o visualizar la ventana de diálogo de la seguridad (si Ctrl-Alt-Del), etc.

b) Entradas y Salidas en Unix

El sistema de entrada/salida se divide en dos sistemas complementarios: el estructurado por bloques y el estructurado por caracteres. El primero se usa para manejar cintas y discos magnéticos, y emplea bloques de tamaño fijo (512 o 1024 bytes) para leer o escribir. El segundo se utiliza para atender a las terminales, líneas de comunicación e impresoras, y funciona byte por byte.

En general, el sistema Unix emplea programas especiales (escritos en C) conocidos como drivers (manejadores) para atender a cada familia de dispositivos de E/S. Los procesos se comunican con los dispositivos mediante llamadas a su driver. Además, desde el punto de vista de los procesos, los drivers aparecen como si fueran archivos en los que se lee o escribe; con esto se logra gran homogeneidad y elegancia en el diseño.

Cada dispositivo se estructura internamente mediante descriptores llamados número mayor, número menor y clase (de bloque o de caracteres). Para cada clase hay un conjunto de entradas, en una tabla, que aporta a los drivers de los dispositivos. El número mayor se usa para asignar manejador, correspondiente a una familia de dispositivos; el menor pasa al driver como un argumento, y éste lo emplea para tener acceso a uno de varios dispositivos físicos semejantes.

Las rutinas que el sistema emplea para ejecutar operaciones de E/S están diseñadas para eliminar las diferencias entre los dispositivos y los tipos de acceso. No existe distinción entre acceso aleatorio y secuencial, ni hay un tamaño de registro lógico impuesto por el sistema. El tamaño de un archivo ordinario está determinado por el número de bytes escritos en él; no es necesario predeterminar el tamaño de un archivo.

El sistema mantiene una lista de áreas de almacenamiento temporal (buffers), asignadas a los dispositivos de bloques. El Kernel usa estos buffers con el objeto de reducir el tráfico de E/S. Cuando un programa solicita una transferencia, se busca primero en los buffers internos para ver si el bloque que se requiere ya se encuentra en la memoria central (como resultado de una operación de lectura anterior). Si es así, entonces no será necesario realizar la operación física de entrada o salida.

Existe todo un mecanismo de manipulación interna de buffers (y otro de manejo de listas de bytes), necesario para controlar el flujo de datos entre los dispositivos de bloques (y de caracteres) y los programas que los requieren.

Por último, y debido a que los Drivers de los dispositivos son programas escritos en lenguaje C, es relativamente fácil reconfigurar el sistema para ampliar o eliminar dispositivos de E/S en la computadora, así como para incluir tipos nuevos.

c) Sistema operativo Linux

Introducción:

El uso creciente de sistemas operativos de propósito general, como Linux, para dar soporte a aplicaciones de propósito específico, como las de multimedia, vídeo bajo demanda o tiempo real no crítico, hace necesario introducir en dichos sistemas operativos mecanismos de control de la entrada/salida que permitan asignar los recursos de forma adecuada. Entre estos mecanismos, el sistema de ficheros y los planificadores de disco son elementos clave a modificar.

Sistema de I/O en linux:

Linux reconoce cinco tipos de dispositivos:

- Dispositivos de discos
- Dispositivos de cinta
- Terminales
- Líneas de comunicación
- Impresoras

Cada dispositivo de E/S está asociado a un archivo especial. Estos archivos se encuentran dentro del directorio /dev y son creados por medio del systemcall *mknod*.

Los archivos son manejados por el sistema de archivos y para escribir y leer en esos dispositivos, el sistema de archivos realiza escrituras y lecturas tal como si se tratase de un archivo de datos del usuario. Esta característica brinda una interfase sencilla y clara entre los usuarios y los procesos. Para leer y escribir en un dispositivo se debe pedir leer o escribir en su archivo especial asociado. El subsistema de archivos administra los archivos en los dispositivos de almacenamiento, además de establecer una interfase entre los procesos y los dispositivos.

Los archivos especiales tienen asignados dos números conocidos como el número mayor y el número menor. El número mayor referencia a qué controlador responde, mientras que el número menor es un parámetro que se le envía al controlador correspondiente indicado por el número mayor.

Los dispositivos de comunicaciones o de red son accedidos sólo mediante funciones de intercomunicación de procesos y no tienen ningún archivo especial asignado ya que la interfase *raw-socket* brinda una interfase más natural para éste tipo de dispositivos.

Algunos de los dispositivos más importantes son:

/dev/mem Dispositivo que ejecuta cada acceso a memoria

/dev/null Asigna salida nula a un proceso por ejemplo

/dev/random Crea números al azar. Si un programa requiere generar números al azar podría utilizar este dispositivo en lugar de crear su propio algoritmo.

/dev/urandom Crea números al azar al igual que /dev/random pero con una peor distribución por otro lado es mas rápido que /dev/random.

/dev/ram Brinda acceso a la RAM de disco. Esta es una especie partición que está ubicada en la MP en lugar de en un dispositivo físico. Su contenido es volátil por lo que se perdería si se reinicia o se apaga la computadora.

/dev/fd0 Apunta directamente al primer dispositivo de disco flexible. El segundo disco flexible es el archivo especial /dev/fd1 .

/dev/hda y /dev/hdb Corresponden respectivamente a los discos dispositivos de almacenamiento masivo (discos rígidos o cd-rom) maestro y esclavo respectivamente, que están conectados al primer puerto IDE. El sufijo de números a estos dispositivos indica el número de partición.

/dev/lp0 Dispositivo de impresora conectado al primer puerto paralelo.

/dev/loop0 Pseudo dispositivo de comunicación que le permite a un programa local conectarse con la dirección IP 127.0.0.1

/dev/sda Se utiliza de la misma manera que /dev/hda sólo que en este caso el dispositivo es un disco rigido SCSI (se llegan a soportar un máximo de 16 SCSI dispositivos).

/dev/psaux Dispositivo que obtiene información de un mouse PS/2.

/dev/mouse Dispositivo que obtiene información de un mouse serial.

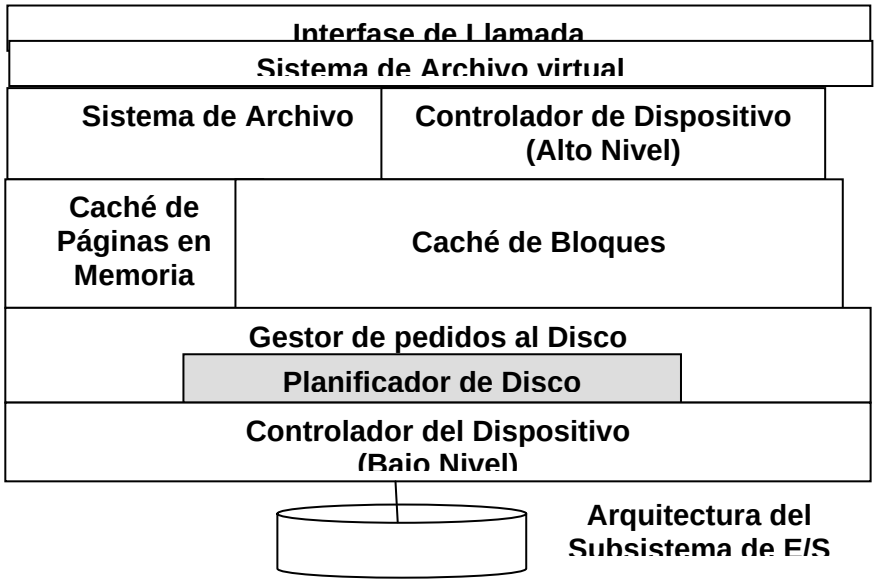
/dev/js0 Dispositivo que se comunica con un joystick analógico.

/dev/parport0 Se puede leer y escribir en el puerto paralelo a través del archivo /dev/ parport

/dev/ttyS0 to /ttyS61 Dispositivos de comunicación conocido como puertos COM1, COM2, COM3 en MS-DOS, en este caso van del 0 al 61.

En la figura se pueden distinguir las distintas partes que lo componen. Entre estas partes hay que destacar el gestor de peticiones al disco y el planificador del disco. El gestor de peticiones al disco se encarga de recoger las peticiones de lectura o escritura en forma de bloques de caché listos para transferir datos desde o hacia el disco. Además se encarga de juntar las transferencias del mismo tipo, cuyos bloques sean contiguos, para que se realicen como una sola petición. Cuando tiene una nueva petición, llama al planificador de disco para que éste la introduzca de forma ordenada en la cola de peticiones. En Linux existe una cola de peticiones por cada tipo de dispositivo. El planificador instalado en Linux por

defecto implementa una política de planificación de tipo CSCAN. Para poder diseñar un sistema que cumpla los objetivos previamente marcados hay que resolver dos problemas fundamentales:



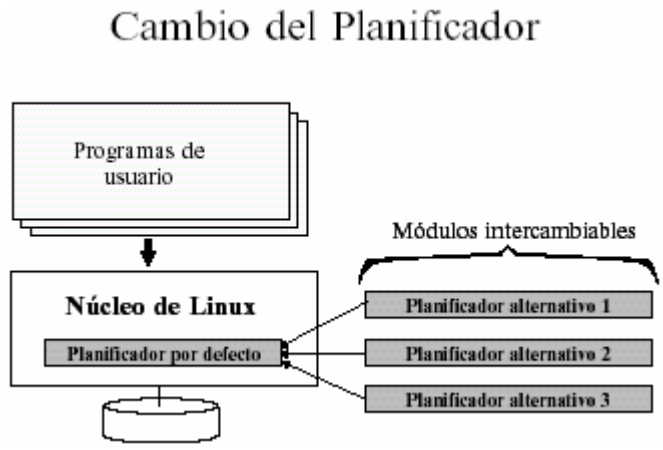
Arquitectura del Subsistema de E/S

- Descubrir un método que permita cambiar de planificador de disco de la forma menos traumática posible.
- Modificar la interfase de E/S de Linux para permitir que las peticiones al disco incluyan parámetros adicionales a los habituales.

Para resolver el problema planteado en el primer punto existen dos posibles soluciones.

- Incorporar el nuevo planificador dentro del código fuente que compone el núcleo principal de Linux.

Esta solución es la más sencilla de implementar. Su principal desventaja, sin embargo, es la dificultad que impone a la hora de incorporar nuevos planificadores de disco al sistema. Para realizar nuevas incorporaciones es necesario recompilar todo el núcleo y reiniciar el sistema completamente. Modificar el núcleo de Linux para permitir la incorporación de nuevos planificadores de disco en forma de módulos externos.



Para conseguir esta funcionalidad nos basamos en la capacidad que tiene Linux desde su versión 2.0.X para incluir parte del código del núcleo en módulos compilados de forma independiente y que luego son instalados en el sistema mientras este está en funcionamiento. Su principal ventaja consiste en poder efectuar todo el proceso de desarrollo de los nuevos planificadores de disco de forma independiente del resto del código de Linux e incluirlos después en el sistema sin tener que reiniciarlo. Para que esto sea posible es necesario modificar el núcleo de Linux para que admita la inclusión de nuevos planificadores a partir de módulos externos.

Implementación del nuevo planificador dentro del código fuente que conforma el núcleo:

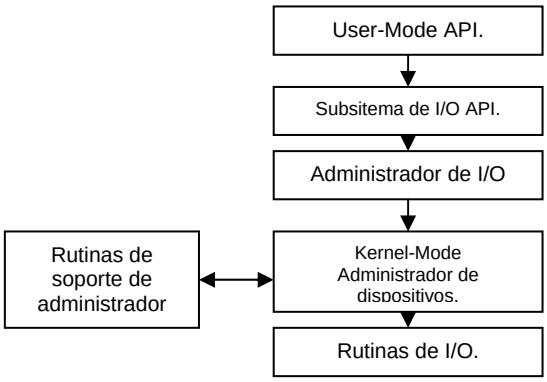
Tal como muestra la figura es posible tener los planificadores de disco desarrollados fuera del sistema e incluirlos sólo cuando sea necesario. Además, y dado que la planificación de las peticiones de E/S es una tarea indispensable del sistema, es necesario incluir en el sistema un planificador de disco por defecto para cuando no se haya especificado ningún otro planificador. Este planificador será el mismo que incluye Linux por defecto, que usa la política CSCAN.

En cuanto al segundo problema a tratar, modificar la interfase de E/S para permitir peticiones al disco con parámetros adicionales, hay que tener en cuenta los siguientes puntos:

- En primer lugar, las peticiones de E/S de los programas de usuario deben ser realizadas través de llamadas al sistema. Para la realización de peticiones con parámetros adicionales se sobrecarga la llamada al sistema ioctl. Esta llamada se usa generalmente para operaciones de control sobre los dispositivos físicos. La interfase de que dispone la llamada ioctl es muy versátil, pudiendo implementarse con ella cualquier tipo de llamada. La idea principal consiste en implementar dos nuevas funcionalidades a la llamada ioctl: una de lectura y otra de escritura. Además de los parámetros de lectura y escritura habituales, estas funciones incorporan los parámetros adicionales necesarios para controlar las nuevas políticas de planificación de disco.

- Otro aspecto a tener en cuenta es cómo pasar los parámetros extra a través de la cache del sistema. La forma más fácil es incluir estos parámetros en la estructura de control de cada bloque de cache. De esta forma, cuando un bloque de cache sea enviado al gestor de peticiones al disco y éste lo pase al planificador de disco, se podrán consultar todos estos parámetros adicionales y realizar la planificación en función de estos datos. Es importante destacar que cuando un bloque de cache está asociado a una petición de E/S sin parámetros adicionales, su estructura por defecto debe incluir un valor por defecto para estos parámetros adicionales.

d) Sistema operativo Windows 2000 *Sistema de I/O en Windows 2000:*



características principales:

- Provee soporte para múltiples sistemas de archivos.
- Permite a los drivers o administradores de dispositivos ser cargador en forma dinámica.
- Conecta aplicaciones y sistemas componentes a dispositivos virtuales, lógicos y físicos.
- Provee una interfase de I/O para cada tipo de dispositivo particular.
- El S.O. abstrae todos los pedidos de I/O como operaciones en un archivo virtual.
    - Escondiendo el hecho que el destino pueda no ser un dispositivo estructurado de archivo.
    - Todos los datos son leídos y recibidos como flujos que tienen como emisor y receptor respectivamente al archivo virtual.
    - El administrador de I/O es el que se encarga de redirigir el conjunto de procesos que involucran al archivo sobre el cual se actúa.

Característica principal de Windows 2000, directorio activo

Introducción:

Directorio activo (DA) lleva la gestión de recursos de Windows NT en el ámbito de la gran empresa fuera de los tiempos pasados: Este servicio elimina los dominios maestros, los dominios de recursos, las cuentas de administrador de acceso total que ponen en peligro la seguridad de un dominio y las numerosas (tantas como N x (N - 1)) relaciones de confianza que había que establecer cada vez que se añadía un dominio nuevo. Para sacar el máximo provecho de DA, es necesario entender por qué es importante un servicio de directorio, cómo trabaja DA, que características proporciona el servicio y cómo afrontar los problemas de migración.

Servicios de directorio:

Un servicio de directorios es un repositorio distribuido de información o punteros a información (usuarios, grupos, recursos, etc). Después de crear un repositorio de este tipo, es posible poner en marcha varias aplicaciones que lo utilicen. Es posible crear aplicaciones sencillas, como un servicio de directorio de información personal o aplicaciones complejas para gestionar un sistema operativo de red (NOS). Al igual que otros servicios de directorio como Novell Directory Services (NDS) o StreetTalk, de Banyan, el DA proporciona un directorio de objetos específicos para el sistema operativo de red que permiten manejar, no sólo los usuarios y sus propiedades, sino también otra gran cantidad de prestaciones específicas de NT, como los volúmenes Dfs, GPO (Group Policy Objects o objetos de directiva de grupos) e infraestructura de claves públicas (Public Key Infraestructure o PKI). El tipo de objetos que puede contener un directorio es

prácticamente ilimitado. Aun así, es necesario tener en cuenta consideraciones de rendimiento y replicación.

El protocolo X.500, basado en la Organización Internacional para la Estandarización (ISO), es, probablemente, el estándar de servicios de directorio más ampliamente conocido. X.500 especifica un esquema por defecto que describe clases de objetos y sus atributos asociados. Los directorios basados en X.500 comparten diversas características. La más importante de los directorios basados en X.500 es la unidad organizacional (Organizational Unit u OU). La OU recibe el apelativo de *objeto contenedor* dentro de un directorio, debido a que la OU puede contener otros objetos (que pueden ser, a su vez, nodos finales u otros contenedores). Dado que los directorios basados en X.500 permiten crear objetos que pueden contener otros objetos, estos directorios permiten el uso de relaciones jerárquicas. Por tanto, es posible relacionar árboles de OU situando a cada árbol como subordinado del anterior. Esta potente posibilidad de DA permite delegar tareas de administración a un subconjunto de usuarios dentro de un dominio W2000. La OU proporciona un control granular de la delegación de la gestión de recursos. En W2000, la unidad de delegación es la OU, mientras que en NT 4.0, este papel lo cumple el dominio.

Otra parte fundamental del servicio de directorio es el esquema, que define la estructura interna de un directorio. El esquema define las relaciones entre las clases de objetos. Cada clase de objeto tiene asociado un conjunto de atributos. Adoptando un modelo orientado a objetos, las clases heredan de otras clases, formando una jerarquía. El esquema del DA es extensible. Esto significa que es posible modificar el esquema para crear nuevas clases y nuevos atributos para las clases existentes.

Características principales:

- Soporta el estándar X.500 para los directorios globales.
- Organización jerárquica que provee un único punto de acceso al sistema de administración (gestión de cuentas de usuarios, clientes, servidores, y aplicaciones, por ejemplo) para reducir la redundancia y los errores.
- Soporte para LDAP.

Mapa de la memoria de E/S en Windows:

La Tabla proporciona un mapa de direcciones de memoria reservadas por el ordenador para los dispositivos periféricos de entrada/salida (I/O).

Dirección	Dispositivo
0000-001F	Controlador de DMA nº 1.
0020-003F	Controlador de interrupciones nº 1.
0040-005F	Temporizadores del sistema.
0060-0060	Controlador del teclado.
0061-0061	Altavoz del sistema.
0062-0062	ACPI-controlador compatible incorporado.
0064-0064	Controlador del teclado.
0066-0066	ACPI-controlador compatible incorporado.
0070-007F	Activación de RTC y NMI.
0080-009F	Registros de páginas DMA.
00A0-00BF	Controlador de interrupciones nº 2.
00C0-00DF	Controlador de DMA nº 2.
00F0-00FF	Coprocesador matemático.
0170-0177	Controlador de la unidad de CD-ROM.
01F0-01F7	Controlador de la unidad de disco duro.
0376-0376	Controlador IDE.
0378-037F	LPT1.
0398-0399	Recursos de la placa base.
03B0-03BB	VGA.
03C0-03DF	VGA.
03E0-03E1	Controlador para tarjetas PC.
03F2-03F5; 03F7-03F7	Controlador de disquete.
03F8-03FF	COM1.

Cuadro comparativo entre Linux y Windows 2000

A continuación se presentará una tabla en la cual se muestran las ventajas que el sistema operativo Windows 2000 ofrece sobre el Linux.
Cabe aclarar que la presente información ha sido rescatada de la página perteneciente a Microsoft.

Requerimiento de Cliente	Linux	Windows 2000 Server
CONSTRUIR SOLUCIONES	• Carencia de controladores de dispensadores de efectivo, touchscreen, lectores de código de barras. • No hay lista de hardware certificado y compatible	• Soporte al hardware más reciente (USB, 1394 Firewire, administración de poder en laptops, etc) • Lista de compatibilidad de hardware detallada • Plataforma integrada construida alrededor de facilidad de uso • Administración basada en scripts para control remoto o local • Pruebas extensivas de aplicaciones y drivers efectuadas por Microsoft
Confiabilidad	• No contiene sistema de archivos con bitácora	• Configuración dinámica (plug and play, hot swap de discos duros) • Certificación de drivers • Protección de escritura sobre el kernel • Sistema de archivos con bitácora para recuperaciones más rápidas • Soporte a encripción y compresión de archivos integrada
Estabilidad	• Soporta 960Mb de RAM. Se debe recompilar para soportar 2Gb de RAM y archivos de 2Gb • Entrada y salida (I/O) sincronía, limitando escalabilidad • Optimizado para hardware muy económico • No hay pruebas TPC-C disponibles referentes a bases de datos	• Windows 2000 soporta 4Gb de RAM, 8Gb en version Advanced y 64Gb en Datacenter • El archivo máximo es 16Tb • I/O Asincrono • Record en TPC-C con los primeros 4 lugares absolutos y los primeros 60 lugares en precio/desempeño • Mejor para servers con dos y cuatro procesadores
Administración	• No hay infraestructura de administración • Bajo nivel de integración de seguridad entre aplicaciones aumenta complejidad de administración	• Administración central incluyendo clusters, directorio, administración delegada y administración basada en políticos • Administración en interfase gráfica o línea de comandos

Interfases de I/O (consideraciones para el diseño)

A continuación se detallará el conjunto de subsistemas que conforman al sistema computacional y que permiten establecer una entre fase entre el procesador y los dispositivos de I/O. Solo se analizarán las unidades de almacenamiento.

Estas interfases son conocidas como controladoras y se puede decir que la función principal que desempeñan es la de permitir una correcta comunicación entre los diferentes elementos o terminales que se encuentren conectados a dicha interfase, esto significa que la misma se debe de encargar de establecer los mecanismos de sincronización y gestión de datos de manera que sea transparente para los elementos conectados a la interfase el acceso a los datos del sistema de I/O del cual se desea extraer datos. Con esto se quiere decir que la interfase será la que se encargue de controlar al dispositivo de I/O para que este pueda recibir o enviar datos, dependiendo de la operación y naturaleza del mismo.

Modos de conexión:

Los dispositivos de almacenamiento deben poder comunicarse con el ordenador para indicarles su presencia y su estado. Para ello han sido definidos varios estándares entre los que destacan

fundamentalmente: SCSI (Small Computer System Interfase), SCSI-2, IDE (Integrated Drive Electronics) y ESDI (Enhanced Small Device Interfase), este último ya en desuso.

SCSI-i:

SCSI es una interfase con una estructura de bus que puede soportar la conexión de diferentes dispositivos, desde discos hasta impresoras y scaners, con un nivel de flexibilidad, capacidad (hasta 3 GB) y de rendimiento muy altos. Cada tarjeta SCSI puede soportar hasta siete dispositivos conectados por un cable de una longitud que puede alcanzar los seis metros, lo que permite que estos dispositivos periféricos puedan estar ubicados fuera del ordenador central. Además, se puede desconectar eléctricamente por sí solos del bus. Su velocidad de transferencia puede alcanzar los 5 MB/s.

Con el estándar SCSI-2 se mejora el rendimiento de su predecesor. La velocidad de transferencia es incrementada hasta los 40 MB/s y el ancho del paquete de datos es de 32 bits, además de incorporar otras facilidades como el almacenamiento duplicado en discos separados.

IDE:

Las ventajas de las interfases IDE son su facilidad de instalación y su precio. En este momento ya es posible disponer de grandes capacidades de almacenamiento. Con esta interfase sólo se pueden controlar dos unidades. Además, cuando se desea instalar un nuevo disco, éste se conecta en daisy-chain al primero, atendiendo a una configuración maestro-esclavo. La velocidad de transferencia que ofrece esta interfase llega hasta los 4 MB/s.

Con la interfase IDE mejorada (evolución del estándar original IDE) la longitud máxima de los cables que conectan los discos con el sistema es de 0.45 m., por lo que únicamente es posible realizar conexiones internas de estos dispositivos. Otra desventaja del IDE es la máxima velocidad de transferencia de datos que es posible alcanzar con esta interfase (10 MB/s, con el IDE mejorado), muy inferior a la que se puede tener con el estándar SCSI-2.

ISA:

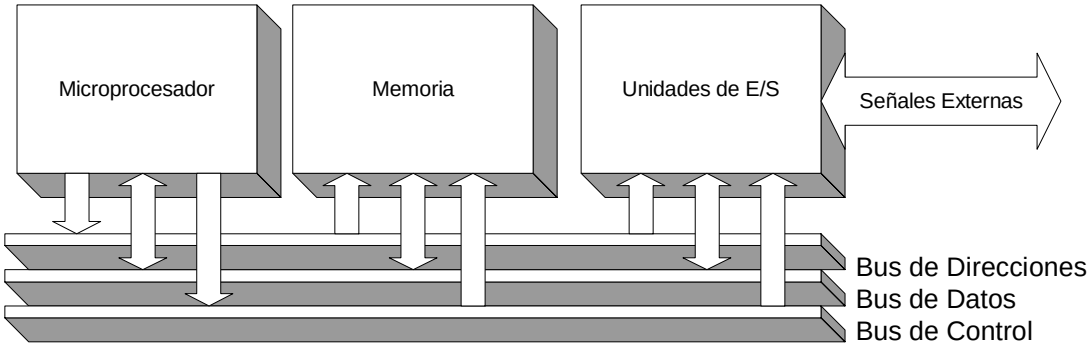
El bus ISA (Industry Standard Architecture) de 16 bits tiene un límite de velocidad teórico de 8 MB/s (en la práctica se reduce a 1,5 MB/s). Esta velocidad presenta el inconveniente de que cualquier unidad IDE actual puede proporcionar más datos que los que el bus ISA puede soportar.

El bus EISA (Extended Industry Standard Architecture) de 32 bits mantiene una velocidad de transferencia de hasta 16 MB/s. Con este bus, los dispositivos SCSI más rápidos tienen que seguir a la espera de mayores prestaciones.

Interfases de Entrada / Salida a nivel de Hardware

Familia de Circuitos que permiten adaptar, leer y / o gobernar señales externas desde y hacia un sistema microprocesador.

Buses



El *bus* es el medio de interconexión entre los diferentes subsistemas de un sistema. El procesador y la memoria central se han de comunicar con los dispositivos de entrada/salida, además de estar interconectados entre sí. En definitiva, el *bus* nos permite tener todos los subsistemas conectados sin la necesidad de tener líneas dedicadas para cada uno de ellos.

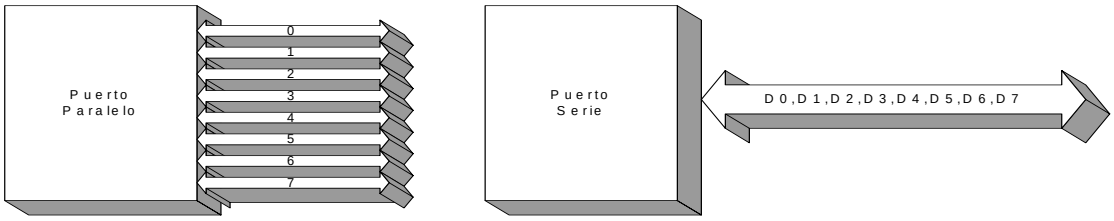
El ***bus*** está compuesto por diversas líneas en paralelo que conectan los diferentes dispositivos. Esta organización permite un bajo coste y versatilidad.

Por otro lado, ofrece una configuración flexible, ya que se pueden ir incorporando nuevos dispositivos sin la obligación de reestructurar el sistema según las nuevas necesidades.
Debido a que por el *bus* pasa toda la información que ha viajar entre los dispositivos, éste se puede convertir en un serio cuello de botella del sistema. Por esta razón, el diseño de *buses* es una pieza clave para aumentar el rendimiento del mismo. Principalmente, las mayores dificultades de diseño son de carácter físico, es decir, la longitud del *bus* y el número de dispositivos interconectados.

Tipos de Entradas / Salidas

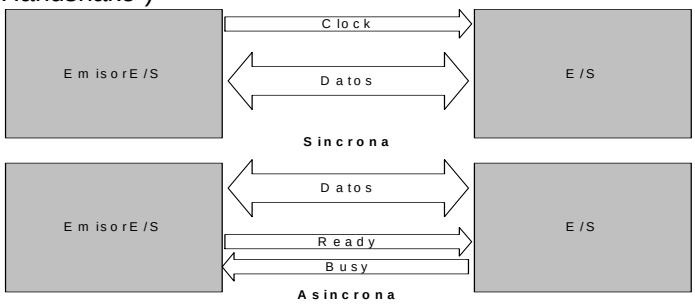
A) FORMATO DE LA INFORMACIÓN:

- Paralelo: una línea por bit del dato y todos simultáneos.
- Serie: todos los datos a través de la misma línea y multiplexados en el tiempo



B) TIPO DE TRANSFERENCIA:

- Sincrónica: se envía o recibe una señal de reloj para sincronizar la transferencia de entrada / salida.
- Asincrónica: no existe señal de reloj de sincronización. Es necesario establecer un protocolo de comunicación (“ Handshake”)



C) TIPO DE SEÑALES ELÉCTRICAS

- Digitales
- Analógicas
- Mixtas

D) DIRECCIÓN DE LOS DATOS:

- Entrada: todas las líneas son permanentemente de entrada
- Salida: líneas permanentemente de salida
- Programables: las líneas son configurables para actuar como entradas o como salidas

E) FUNCIONALIDAD

- Interfaces generales: USART, Puertos paralelo
- Interfaces dedicadas: temporizadores, controlador disco duro,...
- Coprocesadores de E/S

Registros de Entrada / Salida

Son registros asociados a los puertos de E / S.
Tipos de registros:

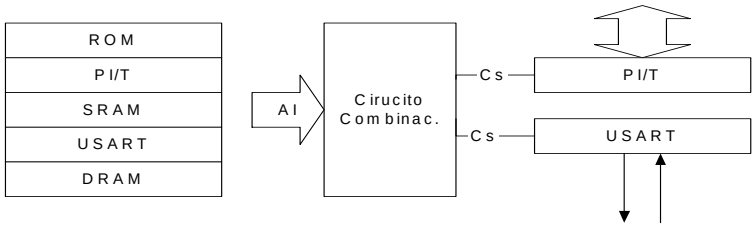
- "Registros de datos: donde se colocan los datos a sacar al exterior o de donde se leen los datos
- "Registros de control: determinan el modo de operación y la configuración de la unidad de E/S

Cada puerto puede tener asociados varios registros
Un mismo registro puede afectar a varios puertos

Pueden ser accesibles mediante operaciones genéricas de lectura / escritura o mediante instrucciones específicas de entrada / salida

Acceso a registros de Entrada / Salida

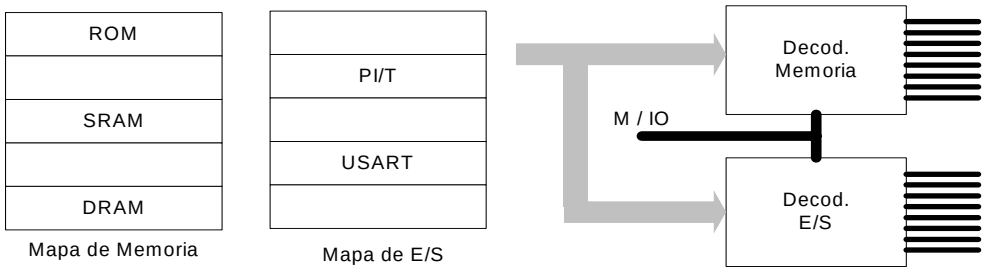
DENTRO DEL MISMO MAPA DE MEMORIA



Las señales de habilitación de los integrados se generan con las líneas del bus de direcciones

CON UN MAPA DE ENTRADAS / SALIDAS SEPARADO DEL MAPA DE MEMORIA:

si el Microprocesador dispone de instrucciones específicas (IN Registro, OUT Registro)



El Microprocesador dispone de salidas que indican si se accede a Memoria o a Registro de E/S. (Salidas del 8086: M / IO, / I O R c, /IOWC) E / S

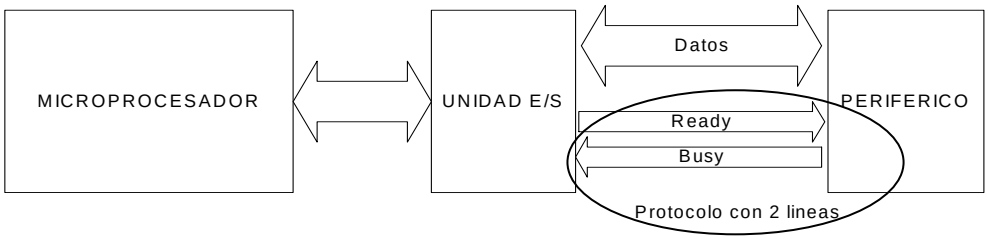
Métodos de Entrada/Salida

BLOQUEO DEL PROCESO:

El Microprocesador espera a que el periférico conectado a la unidad de Entrada/Salida le responda

CONSULTA PERIÓDICA O ESCRUTINIO (POLLING):

El Microprocesador consulta de manera periódica el estado del periférico



INTERRUPCIÓN:

El Microprocesador responde al periférico cuando éste le interrumpe

Métodos de Entrada / Salida

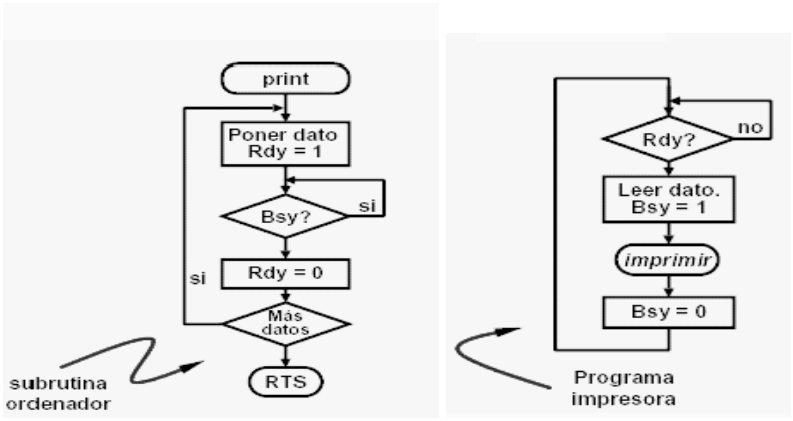
BLOQUEO DEL PROCESO: EJEMPLO IMPRESORA COMO PERIFÉRICO

Ventajas:

- sencillo.

Desventajas:

- Bloquea al microprocesador
- Problemas por diferencia de velocidades



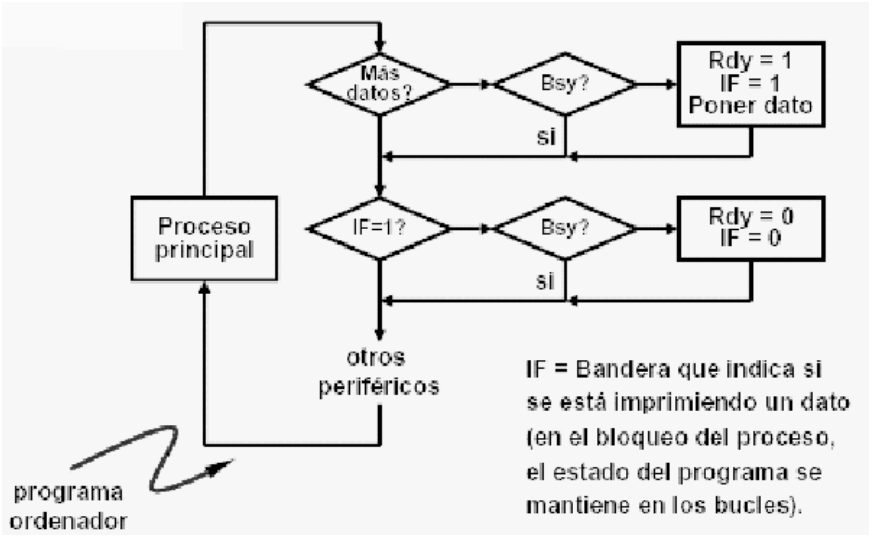
CONSULTA O ESCRUTINIO PERIÓDICA:

Ventajas:

- No detiene el proceso principal

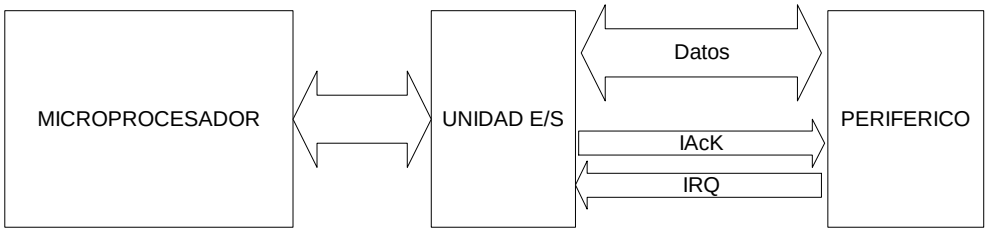
Desventajas:

- Complicado con muchos perifericos



INTERRUPCIÓN:

- IRQ petición de interrupción
- IACK aceptación de interrupción



Desvío del Programa Principal a petición del Periférico (evento)
Se atiende a los periféricos cuando lo requieren y puede la CPU
Secuencia:

- Petición
- Aceptación
- Programa de tratamiento de interrupción

Interrupciones

Los tiempos de ejecución dentro del procesador son muchos más rápidos que los de transferencia de datos a los dispositivos externos a este; por lo tanto si un programa requiere constantemente emitir resultados y solicitar información del exterior, el porcentaje de utilización del procesador sería demasiado bajo. Es por eso que se utilizan las interrupciones, que no es más ni menos que un mecanismo que permite al procesador identificar la ocurrencia de un evento externo y atenderlo.

De esta forma se aprovechan los recursos ya que se evitan los tiempos de respuesta de los dispositivos externos, porque mientras estos están ocupados, el procesador también lo está ejecutando otro programa.

Las interrupciones se procesan de la siguiente manera:

- Un dispositivo externo envía una petición de interrupción (IRQ) al procesador.
- Una vez que el procesador terminó de ejecutar la instrucción en curso, un dispositivo interno del procesador llamado First Level Interrupt Handler (FLIH) chequea hay algún IRQ.
- En ese momento emite una señal de reconocimiento al dispositivo que pidió la interrupción para que esta pueda desactivar la señal.
- Para cargar la rutina de atención a la interrupción, el procesador debe primero salvar el contexto (registros, flags, PC, etc.) en el stack, para que una vez terminada dicha rutina la ejecución del programa continúe en la misma situación.
- Una vez hecho esto el procesador carga el PC con la dirección de memoria que contiene un puntero a la rutina de atención de la interrupción correspondiente. Se carga la dirección en memoria de la primera instrucción de la rutina que atiende a la interrupción y se comienza con su ejecución.
- Cuando finaliza con la rutina, la información guardada en la pila se vuelve a cargar en el procesador y se continúa con la ejecución del programa anterior.

Es un procedimiento algo complejo y que por supuesto incluye algún retraso al tener que ejecutar operaciones extras, pero se puede complicar más si pensamos en varias interrupciones pueden ocurrir al mismo tiempo. Para salvar ese problema se han definido una serie de prioridades que fuerzan a atender a algunas interrupciones antes que a otras; esto es importante porque la velocidad de ciertos dispositivos es bastante elevada y si no son atendidos lo suficientemente rápido, se corre el riesgo de perder información.

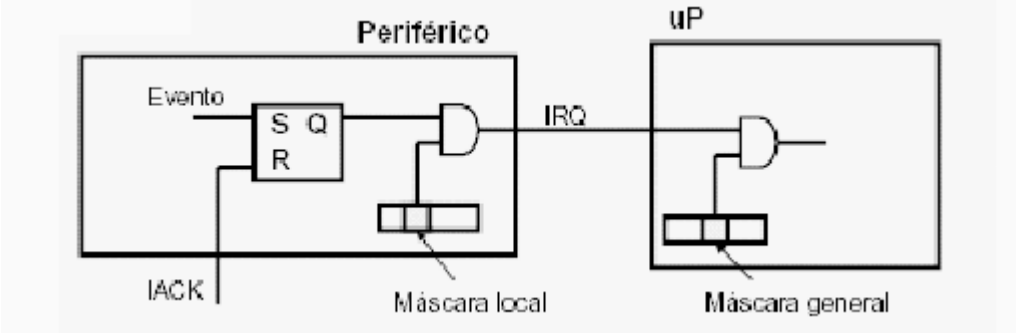
También existe el concepto de interrupciones enmascarables y no enmascarables. Si una interrupción está enmascarada entonces no se atenderá inmediatamente, sino que quedará en espera hasta ser atendida. Generalmente este tipo de interrupciones pertenecen a dispositivos con un bajo nivel de privilegio que no corren peligro de perder información.

Los sistemas operativos utilizan las interrupciones de varias maneras diferentes. Por ejemplo, por medio de una interrupción el sistema operativo detecta una falla, sabe cuando una salida de datos a finalizado, trata excepciones tales división por cero, intento de ejecutar una instrucción privilegiada desde el modo usuario, etc.

También este mecanismo es utilizado cuando se produce un fallo de página y es necesario cargar la página desde memoria secundaria a memoria central. Otro caso es cuando las aplicaciones llaman a las funciones suministradas por el S.O. (System Calls) o también para manejar el flujo dentro del kernel.

ACEPTACIÓN DE INTERRUPCIONES

- Enmascaramiento: Existen interrupciones enmascarables y no enmascarables (se aceptan siempre)
- Las enmascarables se aceptan o no en función del estado de unos bits denominados máscaras que pueden ser locales (del periférico) o globales (generales)



PRIORIDAD DE INTERRUPCIONES

¿Qué sucede si se solicitan varias interrupciones?: Habrá que fijar una prioridad de las mismas
Gestión de la prioridad:

- Prioridad fija (varias líneas de IRQ)
- Prioridad programable
- Prioridad hardware

Controlador de interrupciones específico
Estructura Daisy Chain(Engranajes)

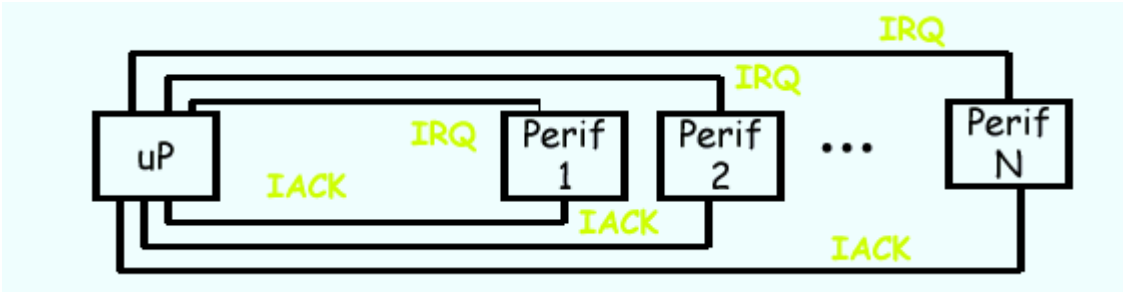
PROGRAMA DE TRATAMIENTO DE LA INTERRUPCIÓN:

Se trata de saber qué dispositivo interrumpe y generar la dirección del programa de tratamiento de la Interrupción

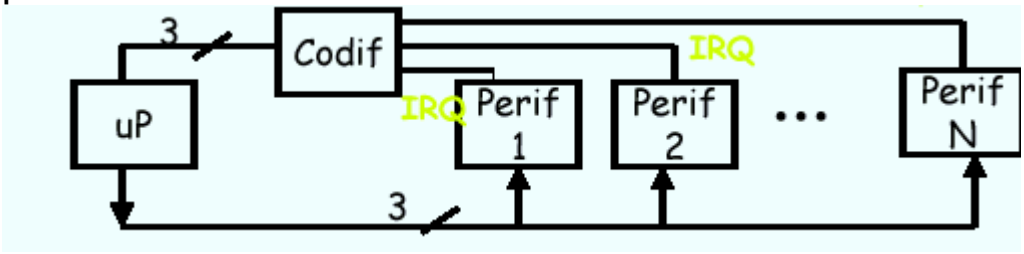
A) Salto a una posición fija para cada línea de petición
B) Tabla de vectores de interrupción:
   a. Autovectorizadas: cada fuente tiene una posición
   b. Vectorizadas externamente por el periférico
C) El periférico genera la dirección de salto.

INTERRUPCIONES CON VARIOS PERIFÉRICOS

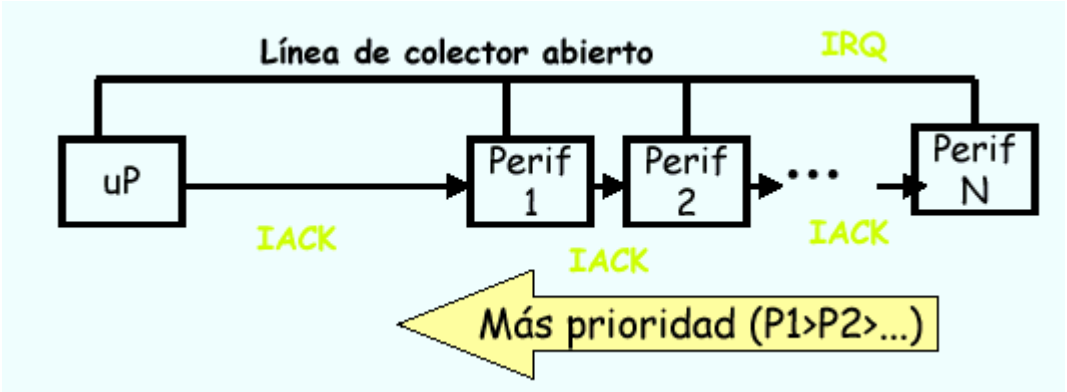
LÍNEAS DE PETICIÓN Y ACEPTACIÓN SEPARADAS PARA CADA PERIFÉRICO



Líneas de petición codificadas

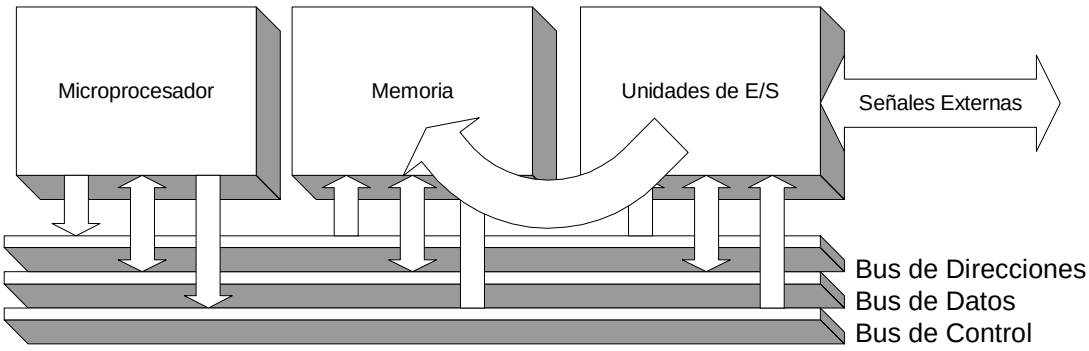


VARIOS PERIFÉRICOS EN UNA SOLA LÍNEA DE PETICIÓN



Acceso Directo a Memoria (DMA)

Consiste en la transferencia directa, sin pasar por la CPU, de datos entre las unidades de E/S y la Memoria del Sistema



La CPU deja de controlar los buses de datos y direcciones durante un tiempo

Aplicaciones:

Cintas y discos magnéticos
Memoria de vídeo para terminales gráficos
Sistemas de adquisición de datos

Necesidad de un controlador de bus:

Controla quién accede en cada momento a la memoria
Posibilidades:
Algoritmo interno a la CPU
Dispositivo externo

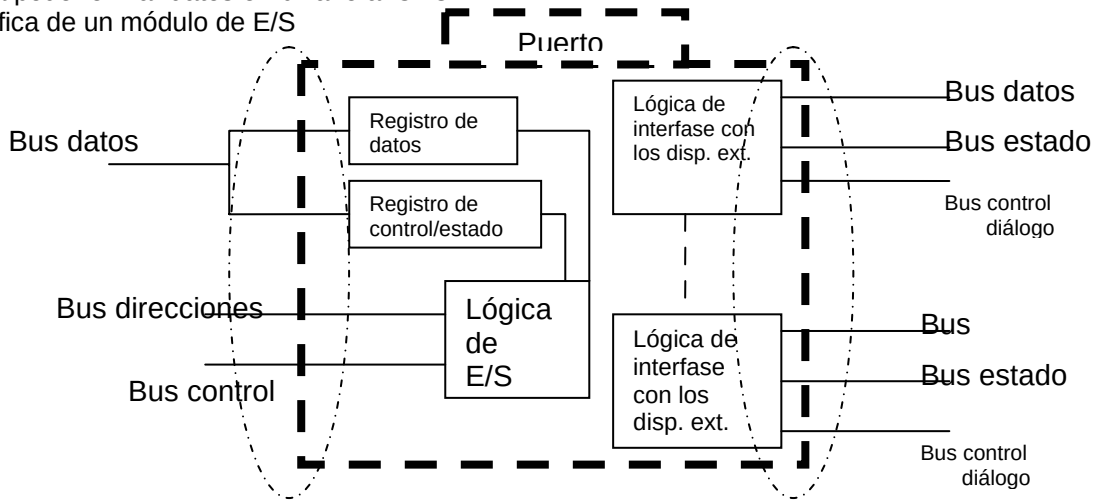
Posibles retardos en la CPU

El módulo de E/S brinda la interfase entre el sistema y el periférico a nivel HW. Está compuesto por registros de datos y de estado o de control.

El módulo de E/S está conectado mediante el bus de sistema al sistema operativo y mediante un puerto al dispositivo periférico. Además está compuesto por una única lógica de E/S a la cual le llega el bus de direcciones y el bus de control y por una lógica de interfase para dispositivo externo a la cual llegan desde el dispositivo un bus de datos, uno de estado y uno de control mediante el cual establece un diálogo generalmente sincronizado.

Contiene una lógica de traducción para tratar los distintos lenguajes a que utilizan los dispositivos para poder enviar datos en binario al CPU.

Gráfica de un módulo de E/S



Consideraciones para el diseño desde el S.O.

- Los dos objetivos principales en el diseño de E/S son: eficiencia y generalidad.
- E/s es extremadamente lenta comparada con la memoria principal
- Uso de multiprogramación permite que algunos procesos esperen e/s mientras otro proceso se está ejecutando
- La e/s no puede mantenerse con la velocidad del procesador
- El intercambio se usa para llevar más procesos listos para ejecución lo cual es una operación de e/s
- La eficiencia es muy importante
- Sería deseable manejar todos los dispositivos de una manera uniforme

- Ocultar la mayoría de los detalles de e/s con dispositivos en rutinas de bajo nivel de forma que los procesos y los niveles superiores del so vean a los dispositivos en términos de funciones generales: read, write, open, y close
- La generalidad es algo importante

Interfase entre el procesador y los dispositivos de entrada/salida

En la mayoría de los programas que se ejecutan en un sistema se producen operaciones de entrada/salida. Ya hemos visto que el procesador se comunica con los dispositivos a través del sistema de *buses*, pero ¿Cómo dialoga con ellos? ¿Qué parte del trabajo de entrada/salida es responsabilidad del procesador y qué parte es de los dispositivos de entrada/salida?
Respondiendo a la primera pregunta, podemos decir que se puede elegir entre dos alternativas para que el procesador direcciones a un dispositivo de entrada/salida. La primera de ellas y la más común se conoce como entrada/salida **mapeada en memoria**. Consiste en reservar una parte del espacio de direcciones de memoria principal para asignársela a los distintos dispositivos. De esta forma, las órdenes que se mandan hacia ellos son accesos a direcciones mapeadas en la memoria. En la otra alternativa, el procesador utiliza códigos de operación de entrada/salida específicos para transmitir las órdenes de entrada/salida. Esto se conoce como E/S aislada. Tanto si se utiliza un esquema como otro, el procesador dialoga con los dispositivos mediante escrituras y lecturas en registros de control y estado que éstos poseen para tal fin. Contestar a la segunda pregunta es más complicado, ya que a lo largo del tiempo se han ido sucediendo varias técnicas que delegan diferentes responsabilidades a las distintas partes del sistema. A medida que los sistemas han ido evolucionando, las técnicas dotaban de más capacidad de proceso a los componentes individuales, aliviando cada vez más al procesador de tener que hacerse cargo de las operaciones de entrada/salida y mejorando el rendimiento. A continuación y para clarificar este punto, describiremos las distintas técnicas o esquemas que se han ido empleando, aunque haremos más hincapié en las que se han desarrollado recientemente.

AUTOEVALUACIÓN DEL MODULO 6:

Preguntas:

1.- ¿De qué tres maneras puede dividirse los dispositivos de E/S según Stallings?

2.- ¿Cuáles son las "Políticas de planificación del disco"? Describirlos brevemente.

3.- ¿Cuáles son los tipos de operación de E/S?

4.- ¿Cómo se relaciona la división en capas con la entrada salida?

5.- ¿Cuáles son los tipos de dispositivos de entrada/salida, referidos al buffer? Desarrollar.

6.- ¿Cuáles son los dos modos de operación de E/S? Describir.

7.- ¿Qué es el caché de disco?

8.- ¿Cuál es la razón para realizar la administración de Entrada/Salida?

9.- ¿Cuál es su funcionalidad de la administración de entrada/salida?

10.- ¿Qué es un modulo de Entrada/Salida?

11.- ¿Que son los dispositivos externos?

12.- Explique la función del DMA

13.- ¿Para que sirve el software de entrada/salida?

14.-De acuerdo a los parámetros de performance del disco, ¿cómo se calcula el tiempo total de transferencia desde el disco?

Múltiple Choice:

1.- **Las razones del porqué los periféricos no se conectan en forma directa al bus del sistema son…** a) Por la variedad de periféricos con varios métodos de operación. b) La velocidad de transferencia de datos de los periféricos es menor que la de la memoria central o del procesador. c) Los periféricos emplean formatos de datos y longitudes de palabra distintos que los utilizados en el computador al que se conecta. d) Porque los periféricos siempre funcionan asincrónicamente e) Todas las anteriores son correctas. f) Ninguna las anteriores son correctas.	**2. La política de SCAN de N pasos es utilizando para administrar las colas del brazo de disco y se caracteriza por…** a) Dividir la cola de peticiones de disco en subcolas de longitud N b) Ser un procesamiento una a la vez mediante un SCAN para cada subcolas c) Que todas las peticiones están en una de las colas y la otra permanece vacía d) Que mientras se procesa una cola, se añadirán nuevas peticiones a las otras e) Todas las anteriores son correctas. f) Ninguna las anteriores son correctas.
3.- Las principales características del RAID de nivel 5 son.... a) Pertenece a la categoría de Acceso Independiente a) Trabaja la Redundancia por código de Hamming b) Su Tasa de Pedido de E/S (Lectura/Escritura) es pobre. c) Aplicación Típica es en Alta tasa de Pedido, lectura intensiva, búsqueda de datos d) Todas las anteriores son correctas. e) Ninguna las anteriores son correctas.	**4. Las principales razones del porqué el S.O. debe controlar las operaciones de Entrada / Salida son:** a) Las interfases del Hardware requieren de un Software complejo para controlarlo y usarlo. b) Los Dispositivos Periféricos son recursos compartidos c) El S.O. provee una interfase consistente, uniforme y flexible para todos los Dispositivos. d) Todas las anteriores son correctas. e) Ninguna las anteriores son correctas.
5. Las principales Funciones del Device Driver son... a) Manejar distintos tipos de dispositivos b) Definir las características Lógicas - Físicas del dispositivo que controla	**6.- Un dispositivo periférico de un sistema de computación es...** a) Todo dispositivo que esta vinculado a través de una comunicación efectiva a un computador.

c) Procesar todas las interrupciones del Hardware de E/S o generada por el dispositivo que controla el Driver d) Permitir la ejecución concurrente de operaciones de E/S en forma controlada. e) Todas las anteriores son correctas. f) Ninguna las anteriores son correctas.	b) Un vínculo y un protocolo de comunicaciones normalizado que conecta un dispositivo con el computador. c) Un computador. d) Un ente lógico externo conectado al computador. e) Un ente físico externo al computador. f) Todas las anteriores. g) Ninguna de las anteriores.
7.- Un BUFFER se usa para.... a) Virtualizar dispositivos. b) Ahorrar espacio en cintas. c) Amortiguar diferencias de velocidades. d) Sincronizar procesos concurrentes. e) Almacenar mensajes de error emitidos por el sistema. f) Interfasear dos dispositivos. g) Ninguna de las anteriores.	**8.- Los dispositivos virtuales son dispositivos …** a) Que permiten transportar un dado software de un equipo a otro con un mínimo de modificaciones. b) Que permiten extender la memoria central de un computador de manera que sea transparente al usuario dicha extensión. c) Hipotéticos que simulan por software un lenguaje de maquina mas conveniente al usuario. d) Hipotéticos que emulan por un código universal de comandos de alto nivel. e) Hipotéticos que simulan mayor cantidad de dispositivos que los físicamente disponibles. f) Todas las anteriores son correctas. g) Ninguna las anteriores son correctas.
9.- El handler de dispositivos es … a) Un procedimiento. b) Un modulo del administrador de e/s. c) Un hardware. d) Una estrategia de gobierno de e/s. e) una modalidad operativa. f) Todas las anteriores son correctas. g) Ninguna las anteriores son correctas.	**10.- El l tiempo de latencia de un disco es el tiempo…** a) Necesario para que el cabezal se coloque en el cilindro adecuado. b) De espera a que la pista alcance la posición del cabezal. c) Que emplea el disco en realizar una vuelta completa. d) Que le toma al disco hacer media revolución e) La Tasa en la cual los datos pueden ser leídos o escritos en la unidad f) Ninguna es correcta.
11.- El planificador del brazo de un sistema de discos... a) Puede estar implementado en el controlador del dispositivo. b) Puede ser escritos como un módulo separado del sistema operativo. c) Debe estar integrado en el Device Driver del disco. d) La implementación del planificador depende de la geometría del disco (cilindros, cabezas y sectores). e) Intenta maximizar el tiempo de servicio. f) Ninguna es correcta.	**12.- Cuál de las siguientes funciones pertenecen a las funciones del software de reloj…** a) Mantener la fecha y hora del sistema b) Generar una interrupción cuando el registro contador llegue a 0. c) Contabilizar el uso del procesador por los procesos. d) Evitar que algunos procesos se ejecuten mas de lo asignado e) Contabilizar el uso de la CPU f) Manejar las llamadas de sistema de ALARMA hechas por los programas del usuario g) Ninguna es correcta.
13.- En el algoritmo de planificación de disco C-SCAN… a) La cabeza de escritura / lectura empieza en un extremo del disco y se mueve hacia el otro, sirviendo las solicitudes a medida que llega a cada pista, hasta que se encuentra en el otro extremo del disco, momento en el cual se invierte la dirección del movimiento y continúa el servicio. b) La cabeza de escritura / lectura se mueve de un extremo del disco al otro, sirviendo las solicitudes a medida que las alcanza y cuando llega al otro extremo vuelve inmediatamente al principio del disco sin servir ninguna solicitud durante el retorno. c) La cabeza de escritura / lectura se mueve como máximo sólo hasta la última solicitud en cada dirección y en cuanto no quedan solicitudes en la dirección actual el movimiento de la cabeza se invierte. el movimiento de la cabeza se invierte. d) La cabeza de escritura / lectura se mueve de un extremo del disco al otro, sirviendo las solicitudes a medida que las alcanza y cuando llega al último pedido vuelve inmediatamente al mínimo pedido al principio del disco sin servir ninguna solicitud durante el retorno. e) Todas las anteriores son ciertas. f) Ninguna de las anteriores es cierta.	14.- **En la Técnica de E/S Programada…** a) Para ejecutar una instrucción del tipo de E/S, el procesador emite una dirección, especificando el módulo en particular y el dispositivo externo, y un comando de E/S. b) El procesador emite un comando de E/S, continúa con la ejecución de otras instrucciones, y es interrumpida por el módulo de E/S cuando éste ha completado su trabajo. c) Hay una estrecha correspondencia entre las instrucciones del tipo E/S que el procesador obtiene (fetch) de la memoria y los comandos de E/S que el procesador emite al módulo de E/S para ejecutar las instrucciones. d) El módulo de E/S y la memoria central intercambian datos directamente, sin involucrar a la CPU. e) Cada palabra de datos que vaya de la memoria al módulo de E/S o al revés debe pasar por la CPU porque es la responsable de la transferencia de los datos. f) Ninguna es correcta.
15.- Algunas funciones que realiza un Device Driver de un dispositivo son: a) Definir las características Físicas del dispositivo que controla. b) Definir las características Lógicas del dispositivo que controla. c) Definir las características Lógicas - Físicas del dispositivo que controla. d) Habilitar y deshabilitar el dispositivo para un dado proceso. e) Permitir la ejecución concurrente de operaciones de E/S en forma controlada. f) Generar todas las interrupciones del Hardware de E/S . g) Todas las anteriores son ciertas. h) Ninguna de las anteriores es cierta.	**16.- Los Manejadores de Interrupciones (Interrupt handler) se ocupa de…** a) error de programación de programa del usuario b) error de checksum transitorio c) error de checksum permanente d) error de búsqueda e) error de controlador f) Todas las anteriores g) Ninguna de las anteriores
17.- El Caché de disco implementado en la controladora o en	**18.- Cuando el procesador desea leer o escribir un bloque de**

la unidad tiene las siguientes características… a. Es rapidísimo b. No usa Memoria Central c. No usa tiempo del microprocesador d. Depende del disco e. Usa muchos discos f. Todas las anteriores son ciertas.	**datos en la E/S emite un comando al módulo de DMA enviándole la siguiente información….** a) Si la operación solicitada es una lectura o una escritura. b) La dirección del dispositivo de E/S involucrado. c) La posición inicial en memoria a leer o a escribir. d) El número de palabras a ser leído o escrito. e) Todas las anteriores son ciertas. f) Ninguna de las anteriores es cierta.
19.- **Un Buffer Circular se caracteriza por…** a) El almacenamiento permanente que soluciona los problemas de "cuellos de botella" en la demanda de E/S. b) Que se implementa mediante un único buffer en el cual se va guardando la información a medida que es transferida. c) Los procesos transfieren datos desde/hacia un buffer mientras el sistema operativo llena o vacía (según corresponda) el otro buffer d) Que se utiliza si el proceso realiza frecuentes ráfagas de E/S lo que se hace es permitir que el proceso utilice más de dos buffers. e) Que se emplean más de dos buffers. f) Todas las anteriores son ciertas. g) Ninguna de las anteriores es cierta.	**20.- El Software de E/S se caracteriza por…** a) Vincular los distintos módulos para realizar la transferencia de los datos en forma controlada. b) Utilizar una serie de Tablas y un conjunto de rutinas especializadas que se ocupan de organizar, administrar y operar la transferencia de datos. c) Proveer un Sistema de Control Físico - Lógico de E/S d) Organizar el software como una serie de capas que provee un conjunto de niveles de servicios e) Ser escrito de tal manera que pueda ejecutar satisfactoriamente sin importar el periférico sobre el que se opera. f) Todas las anteriores son ciertas. g) Ninguna de las anteriores es cierta.

Respuestas a las preguntas

1. ¿De qué tres maneras puede dividirse los dispositivos de E/S según Stallings?
Los dispositivos de E/S pueden dividirse en los siguientes grupos:

- **Dispositivos legibles por el hombre:** son adecuados para la comunicación usuario – maquina. Ejemplos: impresora, teclado, etc.
- **Dispositivos legibles por la maquina:** son adecuados para la comunicación con el equipo. Ejemplos: discos, censores, etc.
- **Dispositivos de comunicaciones:** son adecuados para la comunicación con dispositivos remotos. Ejemplos: módems, etc.

2. ¿Cuáles son las "Políticas de planificación del disco"? Describirlos brevemente.
Las "Políticas de **planificación** del disco" son:

- FIFO (First In First Out – Primero Entrado Primero Salido)
- Prio (Priority - Prioridad)
- LIFO (Last In First Out – Ultimo Entrado Primero Salido)
- SSTF (Shortest Service Time First – Primero el tiempo mas corto de servicio)
- SCAN
- C-SCAN (Circular SCAN)
- N-step SCAN y FSCAN

3.- ¿Cuáles son los tipos de operación de E/S?

- **Asincrónico.**
- **Sincrónico.**

El modo asincrónico es usado cuando hay posibilidad de optimizar la performance de la aplicación. Con la E/S asincrónica, una aplicación inicia una operación de E/S y luego puede continuar su procesamiento mientras el requisito de E/S es completado.
Con el modo sincrónico de E/S, la aplicación es bloqueada hasta que la operación de E/S se completa.
El modo asincrónico es más eficiente, desde el punto de vista del proceso o hilo llamado, porque permite que el proceso o hilo continúe su ejecución mientras la operación es encolada por el administrador de E/S y subsecuentemente ejecutada. Sin embargo, la aplicación que se invoca con el modo de E/S asincrónico necesita alguna manera de determinar cuando la operación es completada.

4.- ¿Cómo se relaciona la división en capas con la entrada salida?
La filosofía jerárquica establece que el sistema operativo debe ser separado de acuerdo a su complejidad, escala de tiempo y el nivel de abstracción. Si se sigue ese criterio, se tiene que el sistema operativo va a estar divido en capas, donde cada una de las capas realiza una cantidad de funciones determinadas. En el mejor de los casos, las capas deben ser diseñadas de manera tal que los cambios en una de ellas no afecten al resto. Uno de los extremos va a tener relación con el hardware de las computadoras, en tanto que el otro va a tener relación con el usuario.
Aplicando esta filosofía a la entrada/salida llegamos a un determinado tipo de organización. El tipo de organización va a depender del tipo de dispositivos y aplicaciones. Las tres estructuras más importantes son:

- **Dispositivos periféricos locales.**
  - Entrada/Salida lógica*.
  - Dispositivos de entrada/salida*.
  - Planificación y Control*.
- **Puertos de comunicación.**
  - Arquitectura de comunicación*.
  - Dispositivos de entrada/salida*.
  - Planificación y Control*.
- **File system.**
  - Directorios de administración*.
  - File system*.
  - Organización física*.
  - Dispositivos de entrada/salida*.
  - Planificación y Control*.

*Cada una de ellas son capas. Comenzando con las que están relacionadas con los procesos del usuario y las ultimas se relacionan con el hardware.

5.- ¿Cuáles son los tipos de dispositivos de entrada/salida, referidos al buffer? Desarrollar.
Los dos tipos de dispositivos de entrada/salida relacionados con el buffer son:

- Block-oriented.
- Stream-oriented.

Block-oriented (orienta a bloques): Son dispositivos que almacenan información en bloques que son usualmente de una medida determinada, y la transferencia es hecha de a un bloque a la vez. Generalmente, es posible hacer referencia por el número de bloque. Ejemplos: discos, cintas.
Stream-oriented (orientada a flujo): Son dispositivos que transfieren datos como cadenas de bytes, con una estructura que no tiene bloques. Ejemplos: terminales, impresoras, puertos de comunicación, mouse.

6.- ¿Cuáles son los dos modos de operación de E/S? Describir.

Los dos tipos de operación de E/S son:

- **Asincrónico.**
- **Sincrónico.**

El modo asincrónico es usado cuando hay posibilidad de optimizar la performance de la aplicación. Con la E/S asincrónica, una aplicación inicia una operación de E/S y luego puede continuar su procesamiento mientras el requisito de E/S es completado.
Con el modo sincrónico de E/S, la aplicación es bloqueada hasta que la operación de E/S se completa.
El modo asincrónico es más eficiente, desde el punto de vista del hilo llamado, porque permite que el hilo continúe su ejecución mientras la operación es encolada por el administrador de E/S y subsecuentemente ejecutada. Sin embargo, la aplicación que se invoca con el modo de E/S asincrónico necesita alguna manera de determinar cuando la operación es completada.

7.- ¿Qué es el caché de disco?

El termino memoria Caché es usualmente asignado a una memoria que es mas pequeña y rápida que la memoria central y que es interpuesta entre la memoria central y el procesador. Esta memoria caché reduce el promedio del tiempo de acceso a memoria mediante la utilización del principio de “localización” (cuando un bloque de datos es puesto en la caché para satisfacer una E/S es porque habrá futuras referencias a ese bloque).
El mismo principio puede ser utilizado para la memoria de disco. Específicamente, una caché de disco es un buffer en la memoria central para discos de sectores. La caché contiene una copia de algunos sectores en el disco. Cuando se hace un requerimiento de E/S sobre un sector particular, se hace un chequeo para ver si el sector se encuentra en la caché de disco. Si esta, entonces el requerimiento es satisfecho, en caso contrario se lee el sector del disco.

8.- ¿Cuál es la razón para realizar la administración de Entrada/Salida?

Se realiza por la gran cantidad de periféricos diferentes que hay, con distintos modos de operación. Debido a esta gran variedad, se dificulta colocar en el procesador los mecanismos necesarios para controlarlos a todos.

Además de utilizar distintas estructuras de datos que las que usa el procesador. Y cada periférico, tiene una velocidad de transferencia, que es menor a la de la CPU y la memoria central, por lo que la administración de Entrada/Salida proporciona una memoria intermedia para que no cause problemas la diferencia de velocidad existente.

9.- ¿Cuál es su funcionalidad de la administración de entrada/salida?

Las funciones de la administración de entrada/salida son:

- Dar una interfase entre la CPU y la memoria central (a través del bus del sistema).
- Llevar un Registro de estado: Dispositivos fuera de servicio (descompuestos o apagados), en servicio (en uso o sin uso).
- Organizar, administrar y operar los dispositivos y la transferencia de datos.
- Controlar los dispositivos.
- Proveer una interfase entre los distintos dispositivos y el resto del sistema.

10.- ¿Qué es un módulo de Entrada/Salida?

Es una parte del computador, que tiene la responsabilidad de controlar uno o varios dispositivos externos e intercambiar datos entre esos dispositivos y la memoria central y los registros de la CPU. Sus funciones principales, podrían ser, control y temporizacion comunicación con el procesador, comunicación con el dispositivo, buffer de datos, detectar errores y manejar interrupciones de bajo nivel.

11.- ¿Que son los dispositivos externos?

Los dispositivos externos, son los que proveen los medios necesarios para intercambiar datos entre el ambiente externo y el computador. Por ejemplo, los discos (rígidos, magnéticos u ópticos), un monitor, teclado, etc.

¿Qué son los dispositivos internos?

Son los dispositivos que proveen la comunicación en el interior de la maquina (controlada por medio de timers). Ejemplos de estos dispositivos, podrían ser la memoria RAM y los timers.

12.- Explique la función del DMA

El modulo de DMA (Direct Access Memory), tiene la capacidad de simular al procesador y toma el control del sistema desde el procesador para realizar una transferencia de datos con la memoria a través del bus del sistema.

Las distintas formas en que el DMA transfiere los datos son:

- Por ráfagas: El DMA toma el control del bus y lo libera cuando finaliza la transferencia del bloque de datos pedido.
- Por robos de ciclos: El DMA toma el control del bus, transmite una palabra y lo libera (Solo lo retiene por un ciclo).
- DMA Transparente: Solo se realizan las transferencias cuando el procesador no esta utilizando el bus.

- Por demanda: Se toma el control del bus cada vez que se tenga que transferir un bloque de datos, y lo libera cuando no tiene mas datos disponibles para transferir.
- Dato a dato: Se realiza la transferencia de un solo dato y se devuelve el control del bus al procesador. Se repite asi, hasta que se termina de mandar el bloque completo.

13.- ¿Para que sirve el software de entrada/salida?

Sirve para organizar el software en capas, darle al usuario una interfase mas amigable y que para este sea transparente el modo de operar del hardware.

Tiene la función de establecer un vínculo entre los distintos módulos para realizar la transferencia de datos de forma controlada.

14.-De acuerdo a los parámetros de performance del disco, ¿cómo se calcula el tiempo total de transferencia desde el disco?

Para calcular el tiempo total de transferencia es importante tener en claro los siguientes conceptos:

Tiempo de Búsqueda (Seek time): Es el tiempo que se requiere para mover el brazo del disco a la pista solicitada. Desafortunadamente, el tiempo que toma en atravesar no es una función lineal (es decir en función del numero de pistas), pero incluye un tiempo de startup y un tiempo de instalación (es el tiempo después de posicionar la cabeza sobre la pista que se tiene como objetivo hasta que la identificación de la pista es confirmada)

Tiempo de demora de rotación: Los discos magnéticos tienen una velocidad de giro que va desde los 5400 a los 10000 rpm. De esa manera, a los 10000 rpm, el promedio de giro será de 3 ms. Los floppy disk giran, en general, entre los 300 y los 600 rpm. Así el promedio de retraso será entre 100 y 200 ms. Tiempo que tarda el comienzo del sector en llegar a la cabeza.

Tiempo de transferencia: El tiempo de transferencia a o desde un disco depende de la velocidad de giro en la siguiente manera:

$$T = \frac{b}{rN}$$

Donde, T= tiempo de transferencia.
b= numero de bytes a transferir.
N= numero de bytes en la pista.
r= velocidad de rotación, en revoluciones por segundo.

De esta manera, el promedio total de acceso puede ser expresado como

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

T_s es el promedio del tiempo de búsqueda.

Ahora, consideremos los siguientes datos:
“Un disco típico con un promedio de búsqueda de 10 ms, velocidad de giro de 10000 rpm y 512-byte sectores con 320 sectores por pista. Supongamos que se quiere leer un archivo que consiste en 2560 sectores para un total de 1.3 Mbytes.

Entonces,	Tiempo promedio de búsqueda	10 ms
	Tiempo de demora de rotación	3 ms
	320 Sectores leídos	6 ms
		19 ms

Supongamos que las pistas restantes puede ser leídas sin necesidad de tiempo de búsqueda.
Entonces, se tiene que relacionar el tiempo de demora de rotación para cada pista sucesiva. Por lo tanto, cada pista sucesiva es leída en 3 + 6 = 9 ms. Para leer el archivo entero:

Tiempo total: 19 + 7 x 9 = 82 ms = 0.082 segundos.

Respuestas del múltiple choice.

1.- a, b, c.	2.- a, b, d.	3.- a,c.	4.- d.	5.- b, c, d.
6.- a	7.- c, f.	8.- e.	9.- b.	10.- d.
11.- a, b.	12.- a,c, d, e, f.	13.- b.	14.- a,c,e.	15.- c, d, e.
16.- b, c, d, e.	17.- f.	18.- e.	19.- d, e.	20.- f.