Gestión de Entrada/Salida

Estudio de la Entrada/Salida, funciones, objetivos, estructuras de la Entrada/Salida y técnicas para mejorar el rendimiento…

Tabla de Contenidos

Autor: Lina García Cabrera

Copyright: Copyright by Lina García Cabrera

1 Introducción

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S

La parte más **confusa de un sistema operativo es la relacionada con la E/S y el sistema de gestión de ficheros**. Con respecto a la E/S, el **elemento clave es el rendimiento**. La velocidad del procesador continúa aumentando y muchos computadores cuentan con varios procesadores. Las velocidades de acceso a la memoria interna también aumentan pero no tan rápidamente como la velocidad del procesador. Sin embargo, con un uso adecuado y con dos o más niveles de caché interna, el tiempo de acceso a memoria principal puede seguir el ritmo de la velocidad del procesador.

Por otro lado, la E/S es una de las áreas más sórdidas del diseño de los sistemas operativos, ya que es un campo en el que las **generalizaciones son difíciles y abundan las soluciones *ad hoc***. La razón de todo ello es la **gran variedad de dispositivos** empleados. Una determinada configuración puede incluir dispositivos que difieran notablemente en lo que respecta a sus características y modo de operación.

En este módulo se hace una estudio breve de los dispositivos y la **organización de la funciones de E/S**. Luego se examinan los objetivos de diseño y la manera de estructurar las funciones de E/S. Por último, se estudian **técnicas para mejorar el rendimiento** como el *buffering* y el *spooling*.

La **entrada/salida** o I/O, se refiere a la comunicación entre computador (CPU y memoria principal), y el mundo exterior, tal vez un ser humano, u otro sistema de procesamiento de información mediante los dispositivos de E/S (teclado, ratón, monitor, impresora, tarjeta de red, etc.)

OBJETIVOS

- Saber las funciones y objetivos del modulo de E/S
- Conocer cómo se estructura el subsistema de E/S y qué servicios ofrece.
- Comprender cómo se realiza una operación de E/S.

2 Dispositivos de Entrada/Salida

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S
Los dispositivos de E/S se pueden clasificar en:

- **Dispositivos de interfaz humana**: destinados a la comunicación con los usuarios

(teclado, pantalla, ratón, impresora).

- **Dispositivos legibles por la máquina**: apropiados para la comunicación con equipos electrónicos como discos, sensores, controladores e impulsores.
- **Dispositivos de comunicaciones**: apropiados para comunicarse con dispositivos lejanos como tarjetas de red.

Existen grandes diferencias entre clases de dispositivos e incluso pueden ser sustanciales dentro de cada clase. Estos dispositivos pueden diferir en algunos de los aspectos siguientes:

1. **Velocidad.** Puede haber una diferencia de varios órdenes de magnitud entre las velocidades de transmisión de información de diferentes dispositivos. Un disco magnético, por ejemplo, puede transferir 9000 caracteres por segundo, mientras que la velocidad asociada a un teclado de un terminal es de unos pocos caracteres por segundo (¡dependiendo del mecanógrafo!).
2. **Unidad de transferencia**: La información puede transferirse en unidades de caracteres, palabras, bytes, bloques o registros, dependiendo del dispositivo empleado.
3. **Representación de los datos.** Un elemento de información puede codificarse de distintas formas sobre diferentes soportes de E/S. Incluso en un mismo soporte pueden emplearse distintos códigos de caracteres.
4. **Operaciones permitidas**. Los distintos dispositivos difieren en el tipo de operaciones que pueden llevar a cabo. Un ejemplo obvio lo constituye la diferencia entre entrada y salida, otro puede ser la posibilidad que hay de rebobinar una cinta magnética, pero no el papel de una impresora.
5. **Condiciones de error**. El no poder completar con éxito una transferencia de datos puede tener causas tan diversas como un error de paridad, un papel arrugado, sin batería o un error de cheksum, dependiendo del dispositivo que se utilice.

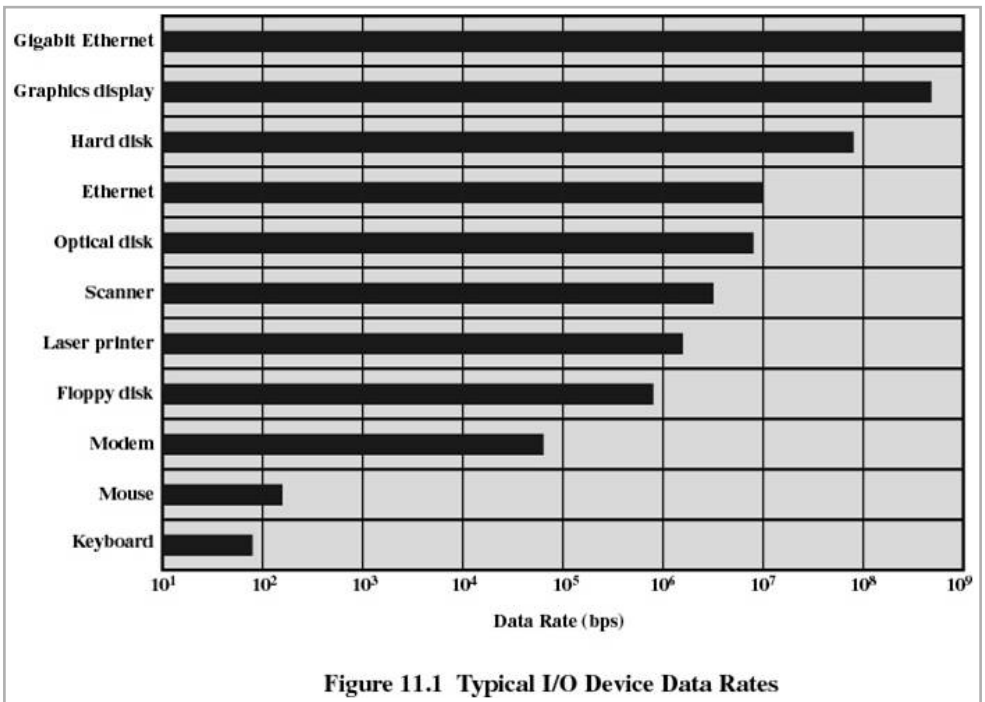


Figure 11.1 Typical I/O Device Data Rates

Figura 1.Velocidades de transferencia de dispositivos (escala logarítmica).

Es evidente que es **difícil lograr un tratamiento uniforme** para la gran diversidad de situaciones vistas en las líneas anteriores. Intentaremos, sin embargo, establecer la **estructura de un sistema de E/S** en el que las características que sean dependientes del dispositivo queden aisladas en lo posible y se logre un **cierto grado de uniformidad**. Empezaremos por considerar algunos objetivos de diseño junto con las implicaciones que llevan asociadas.

3 Objetivos de diseño

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S

1. **Independencia del código de los caracteres**. Evidentemente, es poco deseable que para poder escribir sus programas un usuario deba poseer un conocimiento detallado de los códigos de los caracteres empleados por los diferentes dispositivos. El sistema de E/S debe asumir la responsabilidad de reconocer los diferentes códigos de caracteres y poner de una forma estándar los datos a disposición de los programas de los usuarios.

2. **Independencia del dispositivo**. Hay dos aspectos a considerar con respecto a la independencia del dispositivo.

- En primer lugar, un programa debería ser **independiente del modelo de dispositivo** de un tipo determinado que se le haya asignado. Así, por ejemplo, no debería importar qué impresora se utilice para las salidas de un programa. Una independencia del dispositivo de este tipo asegura que no se malogre un programa simplemente por el hecho de que un dispositivo en particular esté estropeado o bien asignado a algún otro lugar. Se proporciona así al sistema operativo la libertad de asignar un dispositivo del tipo apropiado, de acuerdo con la disponibilidad global del mismo.
- En segundo lugar, aunque esto es más difícil, es de desear que los programas sean **independientes** en todo lo posible, **del tipo de dispositivo** empleado en sus E/S. Evidentemente, la independencia de este tipo no puede llevarse tan lejos como para intentar realizar una entrada desde impresora. Lo que tenemos en mente es la idea de tener que realizar sólo cambios mínimos en un programa para que éste reciba sus datos de un disco en lugar del teclado.

3. **Eficiencia**.Ya que las operaciones de E/S constituyen muy a menudo cuellos de botella dentro de un sistema, es deseable llevarlas a cabo todo lo eficientemente que sea posible.

4. **Tratamiento uniforme de los dispositivo**s. En interés de la simplicidad y de la ausencia de errores es deseable que se puedan gestionar todos los dispositivos de una manera uniforme. Por razones que ya se han mencionado en líneas anteriores, esto puede ser difícil de llevar a la práctica.

Algunas de las consecuencias de estos objetivos son evidentes. En primer lugar, la **independencia del código de los caracteres** implica la existencia de una ***representación interna uniforme*** de todos ellos. Esta representación se denomina el código interno de los caracteres, debiéndose asociar a cada dispositivo los mecanismos adecuados de traducción con el fin de que se lleven a cabo las conversiones adecuadas a las entradas y salidas.

Para los dispositivos que soportan varios códigos de caracteres hay que disponer de distintos mecanismos de traducción para cubrir los distintos códigos soportados. Esta traducción se efectuará inmediatamente después de la entrada e inmediatamente antes de la salida, de forma que los procesos íntimamente ligados a la gestión de los dispositivos deberán conocer el código intermedio empleado. El mecanismo de traducción estará basado generalmente en una tabla, o en algunos casos, en un pequeño fragmento de programa.

Además, la **independencia del dispositivo** implica que los programas deberían trabajar no sobre dispositivos físicos, sino sobre ***dispositivos virtuales*****.** Estos *dispositivos virtuales* se emplean en un programa sin que se haga referencia alguna a los dispositivos físicos: el programador simplemente dirige sus entradas o salidas hacia o desde uno de estos *dispositivos virtuales* en particular.

La **correspondencia *el dispositivo virtual* y tipos de dispositivos para un proceso** en particular puede guardarse en una lista de ***descriptores de***

dispositivos virtuales, a la que apunta el descriptor de proceso o **bloque de control de proceso** correspondiente. La asignación de un modelo particular de un determinado tipo de dispositivo se realiza cuando el proceso utiliza por primera vez el correspondiente *dispositivo virtual*. Diremos que en este momento el proceso abre el *dispositivo virtual*. Este *dispositivo virtual* se cierra, lo que significa que ya no puede ser utilizado más, por acción explícita del proceso, o bien de forma implícita al finalizar éste.

Una tercera consecuencia de los objetivos de diseño que citamos es la de que el sistema de E/S debería construirse de forma que las **características de los dispositivo estuviesen claramente asociadas a los propios dispositivos** en lugar de estarlo a las rutinas que los gestionan (los gestores o **controladores del dispositivo**). De esta forma sería posible que estos controladores de dispositivos mostrasen grandes semejanzas entre ellos y que sus diferentes formas de operación derivasen tan sólo de información de tipo paramétrico, que se pudiese obtener de las características de cada dispositivo en particular. El necesario aislamiento de estas características de los dispositivos puede obtenerse a través de su codificación en sendos ***descriptores de dispositivo*** que se asocien a cada uno de ellos. Estos descriptores constituirán la fuente de información de los diferentes programas de gestión de los dispositivos.

La **información sobre un dispositivo** que puede almacenarse en su **descriptor** es:

1. la identificación del dispositivo.
2. las instrucciones a través de las que actúa.
3. punteros a tablas de traducción de caracteres.
4. el status actual: si está ocupado, libre o estropeado.
5. el proceso de usuario en curso: un puntero al descriptor de proceso (si es que lo hay) que esté utilizando el dispositivo en cuestión.

4 Subsistema de E/S del kernel

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S
El *kernel* debe proporcionar muchos servicios relacionados con la E/S. Servicios como la Planificación de E/S, Almacenamiento en búfer, Almacenamiento en caché, gestión de colas de petición, reserva de dispositivo y Tratamiento de errores.

4.1 Planificación de E/S

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S

Planificar un conjunto de solicitudes de E/S significa determinar un **orden adecuado al ejecutarlas**.

La **planificación mejora el rendimiento global del sistema**, permite **compartir el acceso a los dispositivos** entre los distintos procesos de forma justa y **reduce el tiempo de espera medio requerido para que la E/S se complete**.

Los sistemas operativos incorporan mecanismos de planificación manteniendo una **cola de espera de solicitudes para cada dispositivo**. Cuando una aplicación ejecuta una llamada E/S bloqueante, la solicitud se coloca en la cola correspondiente a dicho

dispositivo. El planificador de E/S **reordena la cola** para mejorar la eficiencia del sistema y el tiempo medio de respuesta. El sistema operativo también puede tratar de ser equitativo o puede dar prioridad a las solicitudes más sensibles.

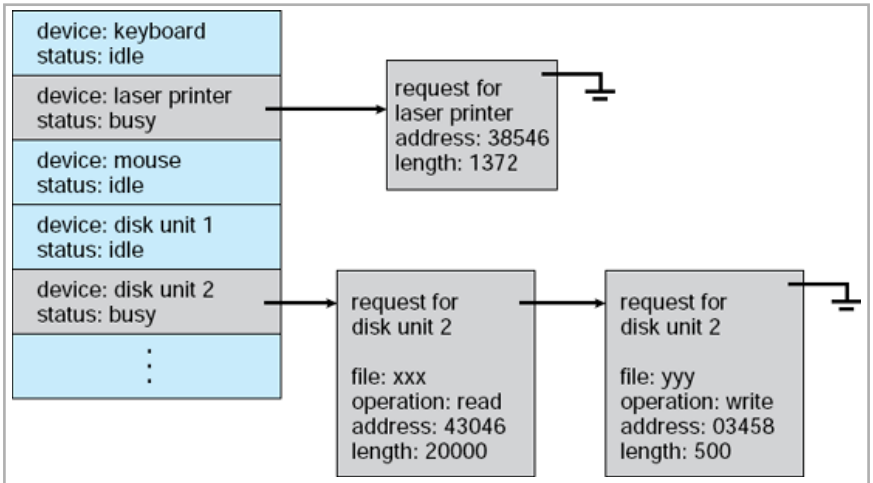


Figura 2.Tabla de dispositivos.

Un kernel con E/S asíncrona controla múltiples solicitudes de E/S al mismo tiempo. Para ello, mantiene una **tabla de estado del dispositivo con una entrada por cada dispositivo**. En cada entrada se guarda información sobre el **tipo, la dirección y el estado** (no funciona, inactivo u ocupado) del dispositivo. Si el dispositivo está ocupado se guarda el tipo de solicitud junto a sus parámetros en la cola del dispositivo correspondiente.

4.2 Almacenamiento en búfer

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S

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

El almacenamiento en búfer se realiza para:

- Adaptar las velocidades entre dos dispositivos con un doble búfer.
- Adaptar dispositivos con diferentes tamaños de transferencia de datos (por ejemplo en la conexión por redes de computadores.
- Permitir lo que se denomina **semántica de copia**. Consiste en copiar los bytes que se quieren escribir o leer a un búfer del kernel para evitar que se sobreescrita la información en el espacio de la aplicación. El mismo efecto se puede lograr de forma más eficiente con mapeo en memoria virtual y con protección de páginas.

4.3 Almacenamiento en caché

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S
Una ***caché es una región de memoria rápida*** que contiene copias de ciertos datos. El acceso a la caché es más rápida que el acceso al original.

La diferencia entre un búfer y una caché es que el **búfer puede contener la única copia de unos datos**, mientras que una **caché**, **almacena en un dispositivo más rápido una**

copia de los datos que están en otro lugar.

Cuando el *kernel* recibe un solicitud de E/S accede primero a la caché para ver si están allí los datos. Asimismo, las escrituras se pueden acumular en una caché para realizar luego grandes transferencias de datos.

4.4 Spooling

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S

Los dispositivos se pueden clasificar en **compartibles**, como los discos, que pueden atender sucesivas peticiones procedentes de diferentes procesos o hilos, y los dispositivos **no compartibles**, tal como las impresoras, que pueden asignarse a un sólo proceso a la vez.

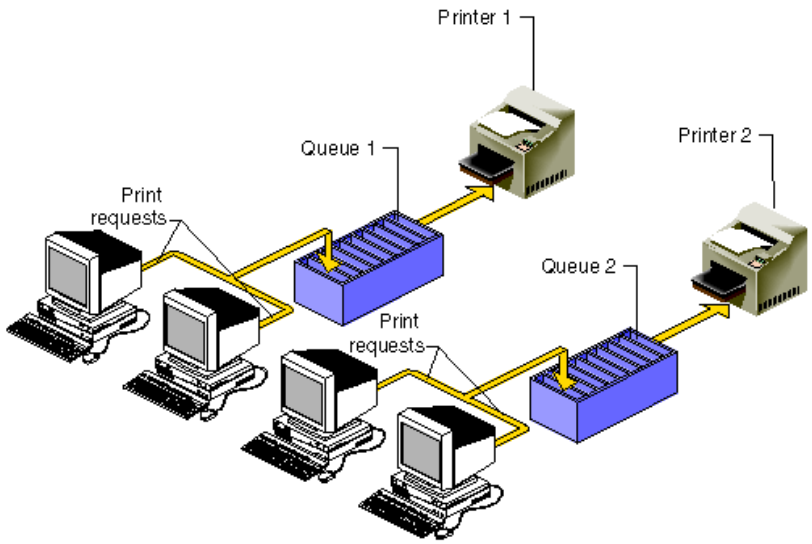
Estos **dispositivos no compartibles** son aquellos que trabajan de forma tal que ***su asignación a diversos procesos a la vez conduciría a una mezcla caótica de operaciones de E/S***. La asignación de un dispositivo no compartible se realiza cuando un proceso abre un *dispositivo virtual* que esté asociado a él. El dispositivo es liberado sólo cuando se cierra el *dispositivo virtual* o bien finaliza el proceso.

Los procesos que quieran utilizar **un dispositivo que ya esté asignado** deben esperar a que sea liberado. Ello implica que durante **períodos de mucha demanda varios procesos pueden quedar bloqueados a la espera** del uso de unos determinados dispositivos, mientras que durante otros períodos de tiempo puede que los mismos dispositivos permanezcan sin ser utilizados. Con el fin de repartir la carga y reducir la posibilidad de cuellos de botella hay que adoptar algún otro tipo de estrategia.

La solución adoptada en varios sistemas consiste en recurrir al ***spooling*** de todas las E/S asociadas a dispositivos no compartibles muy utilizados. Ello significa que en lugar de efectuar la transferencia directamente al dispositivo asociado con un *dispositivo virtual*, **la rutina de E/S lleva a cabo la transferencia sobre un soporte intermedio**, normalmente sobre **disco**.

La responsabilidad de transferir la información del disco al dispositivo en cuestión recae sobre un proceso independiente, denominado ***spooler***, que está asociado con dicho dispositivo.

Como ejemplo,



Spooling de impresora [http://csweb.cs.wfu.edu/~torgerse/Kokua/SGI/007-2861-005/sgi_html/ch02.html]

> consideremos un sistema en el que todas las salidas por impresora se hagan por *spooling*. A cada proceso que abra un *dispositivo virtual* asociado a la impresora se le asignará un fichero anónimo en disco, dirigiendo la rutina de E/S todas las salidas del dispositivo virtual hacia ese fichero que actuará como una impresora virtual. Cuando se cierre el *dispositivo virtual*, este fichero será añadido a una cola de ficheros similares creados por otros procesos, todos ellos esperando a ser impresos. La tarea del *spooler* de la impresora será la de ir cogiendo ficheros de esta cola y mandarlos a la impresora. Se supone, desde luego, que dentro de un período de tiempo determinado, la velocidad de esta impresora será suficiente como para imprimir todos los ficheros generados.

4.5 Tratamiento de errores

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S

Los dispositivos y las transferencias de E/S pueden fallar de muchas formas, debido a razones transitorias (sobrecarga en la red) o permanentes (falla la controladora de disco).

Como regla general, una llamada de E/S al sistema devolverá un **bit de información sobre el estado de la llamada** (si ha tenido éxito o no). En Unix, se utiliza una variable entera denominada ***errno*** para devolver un código de error que indica la naturaleza del error (argumento fuera de rango, puntero erróneo o fichero no abierto).

4.6 Protección de E/S

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S

Los errores están relacionados con cuestiones de protección. Un proceso de usuario no puede ejecutar instrucciones de E/S. Para ello se definen **todas las instrucciones de E/S como instrucciones privilegiadas**. De este modo, la ejecución de instrucciones de E/S se hace solicitando una **llamada al sistema** para que el sistema operativo las realice en **modo núcleo**. El sistema operativo comprueba que es válida y ejecuta la E/S solicitada.

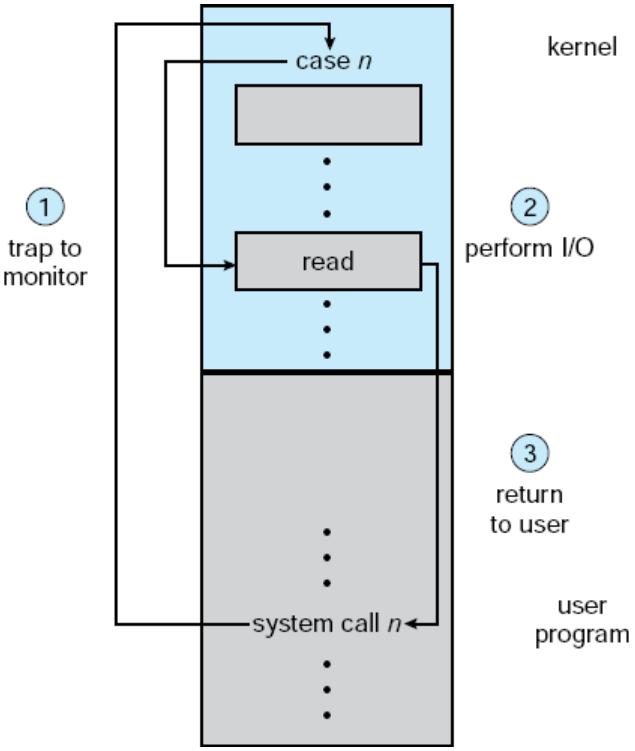


Figura 3. Uso de llamada al sistema para hacer una E/S.

También se debe utilizar el **sistema de protección de memoria** para proteger las **zonas de memoria mapeada y de los puertos de E/S de los accesos de los usuarios**.

El **kernel no puede denegar todos los accesos de los usuarios**. La mayoría de los juegos gráficos y del software de edición y reproducción de vídeo necesita **acceso directo a la memoria de la controladora gráfica** mapeada en memoria para lograr una acceso más rápido. El kernel utiliza en este caso un mecanismo de bloqueo para asignar una cierta zona de la memoria gráfica a un proceso.

4.7 Estructuras de datos del kernel

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S
El sistema operativo mantiene **información de estado sobre el uso de los componentes de E/S**. Para ello utiliza diversas estructuras de datos internas al kernel, como por ejemplo la **tabla de ficheros abiertos**.

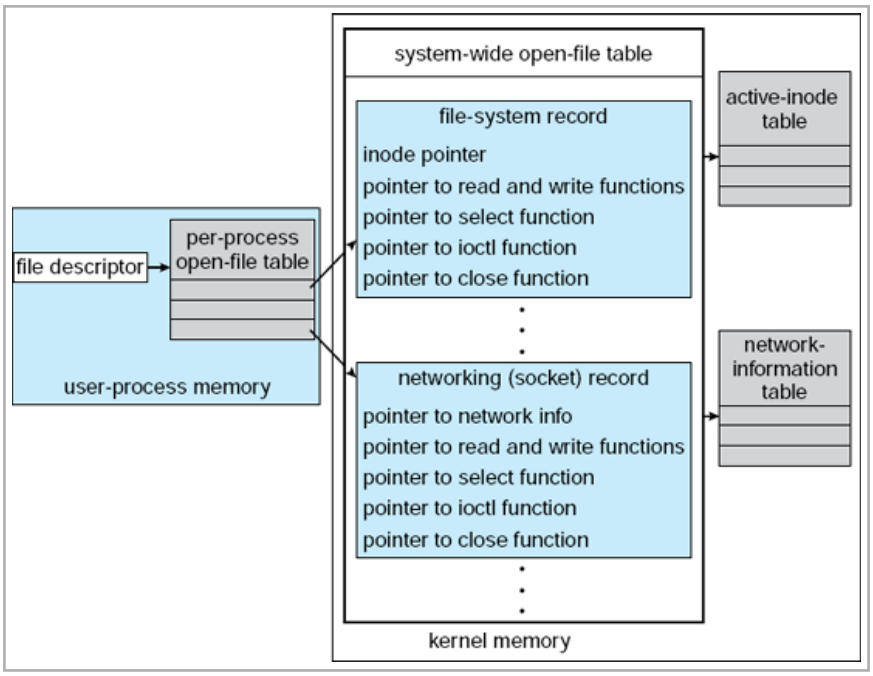


Figura 4. Estructuras de datos E/S en UNIX.

4.8 Servicios del Subsistema de E/S

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S
El subsistema de E/S coordina una gran colección de servicios que utilizan las aplicaciones y otras partes del kernel:

- Gestión del espacio de nombres para ficheros y dispositivos.
- Control de acceso a ficheros y dispositivos.
- Control de las operaciones que admite cada dispositivo.
- Asignación de espacio al sistema de ficheros.
- Asignación de dispositivos.
- Almacenamiento en búfer.
- Almacenamiento en caché y gestión del *spooling*.
- Planificación de E/S.
- Monitorización del estado de los dispositivos, tratamiento de errores y recuperación de fallos.
- Configuración e inicialización de controladores de dispositivos.

Los **niveles superiores del subsistema de E/S** acceden a los dispositivos a través de la **interfaz uniforme** proporcionada por los **controladores del dispositivo**.

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

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Gestión de E/S
¿Cómo consigue el sistema operativo que se realice realmente una operación de E/S por un dispositivo?

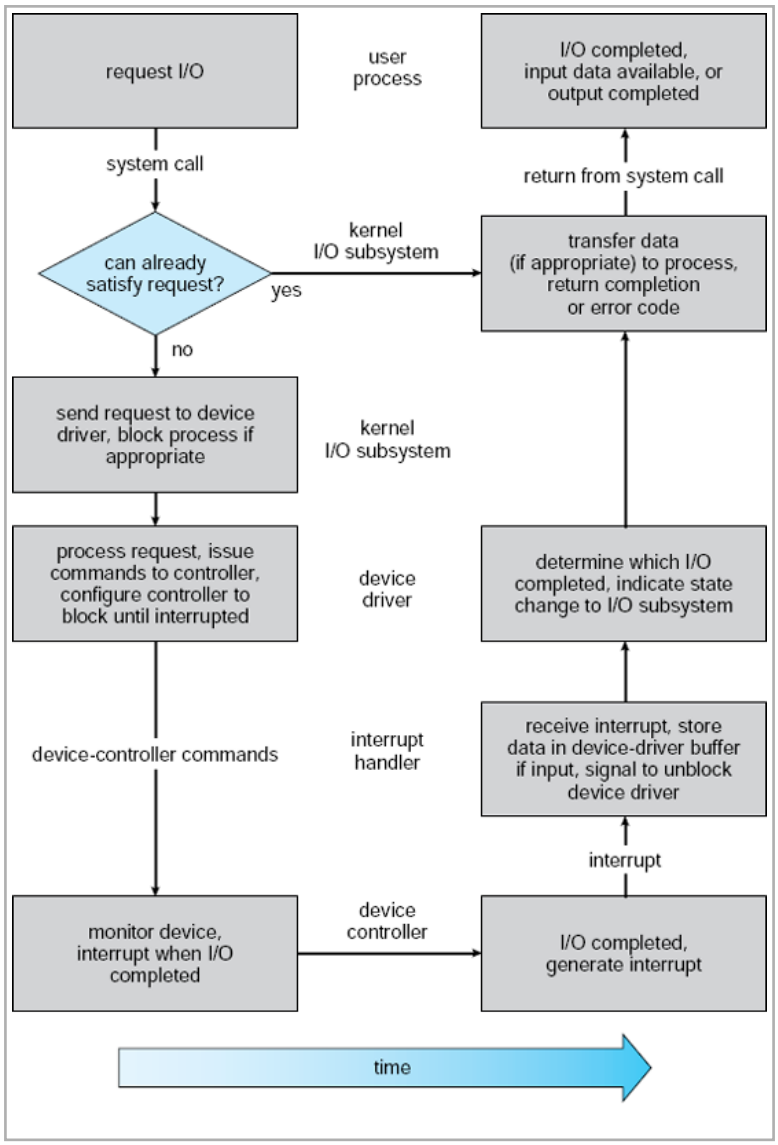


Figura 5. Solicitud de E/S.
Veamos a modo de ejemplo una **solicitud de lectura bloqueante**:

1. **Un proceso ejecuta una llamada al sistema bloqueante** read() dirigida a un **descriptor de fichero** (la estructura de datos con la información del fichero en memoria) o a un fichero abierto con antelación.
2. El código de la **llamada al sistema en el kernel comprueba la corrección de los parámetros**. Si los datos están en la caché o en un búfer, los datos se devuelven al proceso y se completa la solicitud de E/S.
3. En caso contrario, hay que realizar la **E/S física**. El **proceso** (el BCP) se elimina de la cola de preparados y se coloca en la **cola de espera en el dispositivo**, planificándose la solicitud de E/S. El **subsistema de E/S enviará la solicitud al controlador de dispositivo**.
4. El **controlador de dispositivo asigna espacio de búfer del kernel** para recibir los datos y planifica la E/S. El controlador envía las órdenes a la tarjeta controladora del dispositivo escribiendo en los registros del control del dispositivo.
5. El controlador del dispositivo **opera con el *hardware* del dispositivo** para realizar la transferencia de datos.
6. El controlador realiza un sondeo para ver la información de estado y recolectar los datos, o puede haber configurado una transferencia de DMA (acceso directo a memoria) hacia la memoria del kernel. La transferencia la gestiona un controlador de DMA que genera una interrupción cuando la transferencia se complete.
7. La rutina correcta de tratamiento de interrupciones recibirá la interrupción a través de la tabla de vectores de interrupción, almacenará los datos necesarios,

efectuará una señalización al controlador de dispositivo y volverá de la interrupción.

8. El controlador de dispositivo recibe la señal, determina qué solicitud de **E/S se ha completado**, determina el estado de la solicitud y señaliza al subsitema de E/S del kernel que la solicitud se ha completado.
9. **El kernel transfiere los datos** o los códigos de retorno al **espacio de direcciones del proceso** solicitante y mueve el proceso de la cola de espera a la cola de procesos listos o preparados.
10. Cuando el planificador asigne la CPU al proceso, éste reanuda su ejecución en el punto correspondiente a la terminación de la llamada al sistema.