CAPITULO 11

SISTEMAS OPERATIVOS - INTRODUCCION

Un Sistema Operativo es un programa que actúa como interfase entre el usuario de una computadora y el hardware de la misma. El propósito es proveer un entorno en el cual el usuario puede ejecutar programas.

O sea que un objetivo principal de un SO es hacer que un sistema sea CONVENIENTE de usar. Otro objetivo es que se use el hardware de manera EFICIENTE, o sea que está íntimamente ligado a él. Un SO es un programa para un determinado hardware.

Otro objetivo es administrar los recursos de una computadora, tanto de hardware como de software. Por lo tanto, un SO es un conjunto de programas que administran un sistema.

Para entender qué son los SO hay que entender como fueron desarrollándose, lo cual se explica en este apunte.

11.1. ¿ QUE ES UN SO ? [21][S.O.]

Como sabemos, el hardware provee los recursos básicos de un sistema de computación. Mediante los programas de aplicación se definen las formas en que se usan esos recursos para resolver problemas para los usuarios. Hay diversos programas de aplicación. Un sistema operativo controla y coordina el uso del hardware entre los distintos programas de aplicación para los diversos usuarios.

El Sistema Operativo provee los medios para utilizar de forma correcta los recursos de un sistema de computación; no hace ninguna función útil por sí mismo. Simplemente provee un entorno en el cual otros programas pueden hacer trabajo útil.

Podemos ver al SO como un ASIGNADOR DE RECURSOS. Un sistema tiene diversos recursos que pueden requerirse para resolver un problema: tiempo de CPU, espacio de memoria, espacio de almacenamiento de archivos, procesos, dispositivos de E/S, etc. El SO actúa como el gerente de esos recursos y se los otorga a programas específicos y usuarios a medida que son necesarios. Como puede haber diversos pedidos conflictivos para recursos, el SO tiene qué decidir que pedidos son atendidos y cuáles no. Su fuente principal de trabajo es un conjunto de tablas en las cuales se describen los recursos y a que procesos están asignados los mismos.

Otro punto de vista de los SO apunta a la necesidad de controlar los distintos dispositivos de E/S y programas del usuario. Un SO es un programa de control. Un programa de control controla la ejecución de programas de usuario para prevenir errores y uso impropio de la computadora. Está relacionado especialmente con la operación y control de dispositivos de E/S.

Para ver qué son los SO, estudiemos como fueron construidos en los últimos 30 años. Viendo esta evolución, podemos identificar elementos en común, y cómo fueron mejorando.

Los SO y la arquitectura de los computadores tienen una gran influencia el uno sobre el otro. Para facilitar el uso del hardware se construyeron los SO. A medida que se diseñaron y usaron SO, se volvieron obvios ciertos cambios en el diseño del hardware que simplificaron los sistemas operativos. En esta revisión histórica veremos como la introducción de nuevo hardware es la solución natural para muchos de los problemas de sistemas operativos.

11.2. LOS PRIMEROS SISTEMAS[21][S.O.]

Inicialmente solo hubo hardware. Las primeras computadoras eran máquinas muy grandes que se programaban desde una consola. El programador podía escribir un programa, y luego operar el programa directamente desde la consola del operador.

Primero, el programa tenía que cargarse manualmente en la memoria, ya sea por medio de switches (llaves de conmutación), cinta de papel, o tarjetas perforadas. Luego, se apretaban los botones apropiados para cargar la dirección de comienzo y comenzaba la ejecución del programa. A medida que el programa corría, el programador/operador podía monitorear su ejecución por medio de luces en la consola.

Si se descubrían errores, el programador podía parar el programa, examinar los contenidos de la memoria y registros, y corregir el programa directamente desde la consola. La salida se imprimía o perforaba en cintas o tarjetas para una impresión posterior.

Un aspecto importante de este entorno era la naturaleza interactiva HANDS-ON. El programador era el operador. Muchos sistemas usaban un esquema de reserva para otorgar tiempo de maquina: para usar la maquina se pedía un turno en una hoja de papel.

Esta aproximación tiene ciertos problemas: supongamos que uno reserva una hora, y necesita más que ese tiempo para hallar un error. Uno tiene que parar, recolectar lo que se pueda, y volver mas tarde para continuar (consideremos que no existían ensambladores, y mucho menos compiladores). Por otro lado, uno podía terminar antes de que se terminara su tiempo, quedando el resto del tiempo la máquina sin ser usada.

A medida que pasó el tiempo, se construyó software y hardware adicional. Lectoras de tarjeta, impresoras de línea y cintas magnéticas se convirtieron en lo más común. Se diseñaron assemblers, cargadores y linkers para facilitar la programación, así como también bibliotecas de funciones comunes. Estas podían copiarse en un nuevo programa sin tener que escribirse dos veces.

Las rutinas que hacen entrada y salida son especialmente importantes. Cada dispositivo de E/S nuevo tiene sus características propias, requiriendo una programación cuidadosa. Para cada dispositivo se escribe una subrutina especial, llamada el manejador del dispositivo (device driver). Este programa conoce como deben usarse los buffers, banderas, registros, bits de control y estado para un dispositivo en particular. Cada tipo distinto de dispositivo tiene su propio manejador. En vez de escribir el código necesario cada vez, el device driver se utilizaba directamente de la biblioteca.

Posteriormente aparecen compiladores (Fortran, Cobol, etc.) que facilitan la tarea de programación, pero dificultan la tarea de operación de la computadora. Para preparar un programa Fortran para ejecución había que ejecutar los siguientes pasos:

- Montar la cinta magnética o cargar las tarjetas perforadas que contenían el Compilador Fortran.
- Cargar el compilador Fortran.
- Leer el código fuente del programa de tarjetas perforadas y escribirlo en otra cinta magnética.
- Compilar el programa. El compilador Fortran producía una salida en lenguaje ensamblador.
- Montar la cinta que contiene el Assembler.
- Ensamblar el programa.
- Linkeditar la salida del programa para incluir rutinas de bibliotecas y subrutinas.
- Cargar el programa ejecutable y ejecutarlo, para su corrección.

Para ejecutar un trabajo hay una gran cantidad de tiempo de preparación (setup time), debido a la gran cantidad de pasos de que consta el trabajo. Si ocurre un error en algún paso, hay que comenzar nuevamente desde el comienzo, teniendo que cargar cintas o tarjetas.

11.3. MONITOR SIMPLE O SISTEMA BATCH SENCILLO[21][S.O.]

Vimos que el tiempo de preparación es un problema importante. Durante el tiempo de montado de cinta, o mientras el operador trabaja en la consola, la CPU esta desocupada. En esos tiempos, un computador era muy caro, con un tiempo de vida esperado corto. Por lo tanto, el tiempo de computación era muy valioso, y los propietarios querían utilizarlo lo más posible.

Hubo una solución doble: la primera fue contratar operadores profesionales, de tal forma que el programador no operara más la computadora, y eliminar el tiempo desperdiciado por las reservas. Los operadores no saben corregir los programas: lo único que hacen es obtener un vuelco de memoria y registros para el programador, y así poder seguir trabajando.

La segunda solución trató de reducir el tiempo de preparación. Los trabajos con necesidades similares eran loteados (batched) juntos, como un solo grupo. Si el operador recibe varios trabajos Fortran y otros tantos Cobol, en vez de ejecutarlos en orden de llegada ejecuta los programas Fortran juntos, y los Cobol juntos, y se gana el tiempo de carga de los compiladores.

A pesar de mejorar la utilización, sigue habiendo problemas: si para un programa, el operador tiene que mirar la consola para ver la condición de parada, si fue un error, hacer un dump (vuelco de memoria), y luego cargar la lectora con el próximo trabajo. Durante este tiempo, la CPU se mantiene sin uso.

Para solucionar este problema se introduce el secuenciamiento automático de trabajos, y con él, se crean los primeros sistemas operativos rudimentarios. La idea es que un programa pequeño, llamado monitor residente, transfiera automáticamente el control de un trabajo al siguiente.

Inicialmente, el monitor tiene el control del computador. El mismo transfiere el control a un programa. Cuando éste termina, retorna el control al monitor, que dará control al próximo programa. El monitor tiene que saber qué programa ejecutar. Con este fin se introducen las tarjetas de control, para dar información directamente al monitor. Estas son tarjetas especiales que se mezclan con las de datos o programa, y son directivas que indican qué programa se ejecutará y qué recursos necesitará.

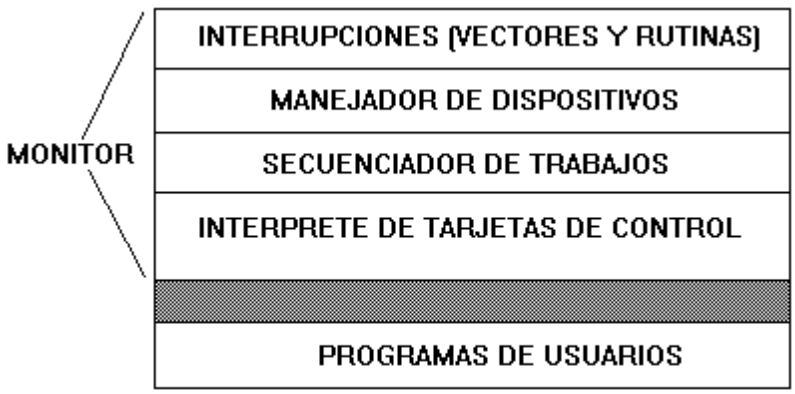

Fig. 11.1. - Esquema de Monitor.

Las tarjetas de control tienen una identificación especial para diferenciarlas de las de datos (//, $, *, ?). P. ej.:

* TRABAJO
* EJECUTAR
* DISCO
* FIN

Los errores son atrapados (interrupciones), y reciben un tratamiento adecuado (por ejemplo, un dump), y luego se sigue ejecutando la próxima tarjeta de control.

Para evitar que un programa, por error, lea tarjetas de control del siguiente trabajo, se utiliza un manejador de dispositivos (o sea que el encargado de leer es el Monitor), y esta instrucción (lectura) es una instrucción privilegiada.

Para hacer un pedido al SO, existen las llamadas al supervisor (SVC en IBM, Trap en PDP, nuestras BSV -bifurcación al supervisor-). Estas provocan una interrupción. En la Fig. 11.2 vemos un ejemplo simple de una lectura.

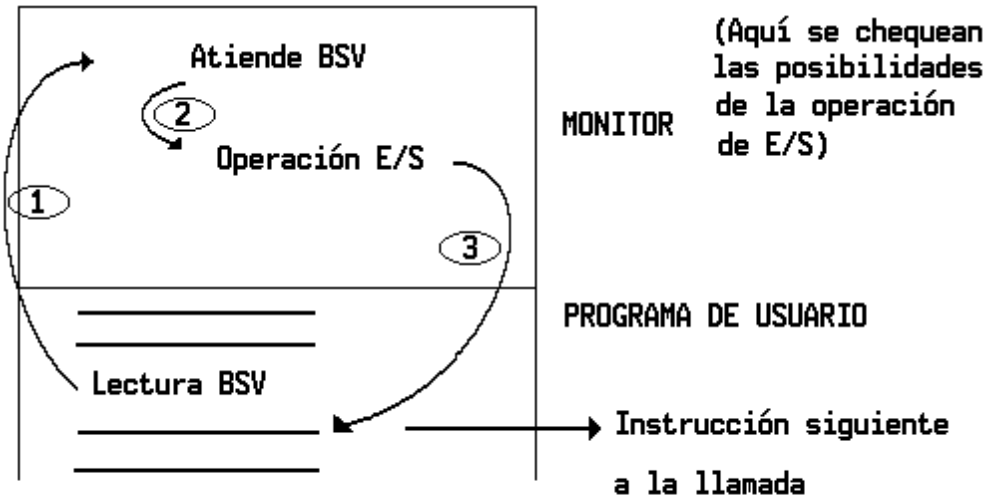

Fig. 11.2. - Operación modo Maestro/Esclavo.

Podemos ver que un monitor residente tiene varias partes identificables.

Una principal es el intérprete de tarjetas de control. Este necesita un programa cargador para cargar los programas y aplicaciones en memoria. El cargador, como el intérprete de tarjetas tienen que hacer E/S. Por lo tanto, el monitor también tiene una serie de manejadores de dispositivos para los dispositivos de E/S del sistema. Para evitar lecturas erróneas, solo el monitor podrá acceder físicamente a la información. Por este motivo surge la necesidad de la existencia de instrucciones privilegiadas (Modo Maestro/Esclavo).

11.4. BATCH SOFISTICADO (PERFORMANCE) [21][S.O.]

La aparición de sistemas batch mejoran la utilización de los sistemas, y mejoran la performance y el rendimiento. A pesar que el secuenciamiento automático elimina la manipulación humana, que es muy lenta, la CPU sigue estando poco utilizada. El problema reside en los dispositivos de E/S, que al ser mecánicos son mucho más lentos que la CPU.

Por ejemplo, un assembler puede procesar 300 o más tarjetas por segundo, mientras que una lectora muy veloz puede leer solo 1200 tarjetas por minuto. Entonces, para ensamblar un programa de 1579 tarjetas, se tardan 4.8 segundos de ensamblado, mientras que 78.9 segundos se ocupan en leer tarjetas. El procesador esperó 74.1 segundos, o sea, un desperdicio del 93.9%.

El mismo problema ocurre para las operaciones de salida. El problema es que mientras ocurre una operación de E/S, la CPU está desocupada esperando que termine la E/S, y mientras la CPU está ejecutando, los dispositivos de E/S están libres.

11.4.1. Operación off-line [21][S.O.]

Una solución para el problema anterior fue reemplazar las lectoras de tarjetas e impresoras, con unidades de cinta magnética. En vez que la CPU leyera directamente de las tarjetas, las mismas primero se copiaban en una cinta magnética. Cuando ésta estaba suficientemente llena, se llevaba a la computadora. Cuando un programa quería leer una tarjeta, lee la cinta. Similarmente, la salida se hace en una cinta para imprimirse posteriormente. Las lectoras e impresoras operaban off-line, no en la computadora principal.

La ventaja principal es que la CPU se libera de la espera de lectura de tarjetas, que son muy lentas, y la misma se hace por medio de cintas, que son mucho más rápidas. Además no son necesarios cambios en los programas de aplicación: los mismos llaman al driver de la lectora de tarjetas, que se reemplaza por un driver de lectura de cinta magnética. De esta forma, los programas de aplicación no deben ser cambiados: solo el driver. Los programas usan dispositivos de E/S lógicos.

La ventaja real en la operación off-line es la posibilidad de usar múltiples lectoras-escritoras magnéticas para una sola CPU. Si la CPU puede procesar entrada a dos veces la velocidad de la cinta, entonces dos lectoras trabajando simultáneamente pueden producir suficiente cinta para mantener a la CPU ocupada.

11.4.2. Buffering[21][S.O.]

Esta es otra solución a la lentitud de los dispositivos de E/S. La idea es muy simple: después que se leyó un dato, y la CPU está operando en el mismo, el dispositivo de entrada comienza a leer el próximo dato inmediatamente. La CPU y el dispositivo están ocupados simultáneamente. Un buffering similar se puede hacer para la salida de datos.

La unidad natural de datos es el registro. Puede ser un registro físico (una línea de impresión, un caracter del teclado, o un bloque de una cinta), o un registro lógico (una línea de entrada, una palabra, o un arreglo). Los registros lógicos están definidos por la aplicación; los físicos, por la naturaleza del dispositivo de E/S. Los registros son las unidades de datos usados para el buffering.

En la práctica, es difícil que el buffering pueda mantener ocupados simultáneamente a la CPU y a los dispositivos de E/S. Si la CPU esta trabajando en un registro, mientras que un dispositivo está trabajando con otro, o la CPU o el dispositivo pueden terminar primero.

Si la CPU termina primero, tiene que esperar, porque no puede procesar otro registro hasta que el dispositivo no haya terminado con el anterior. Si el dispositivo termina primero, o espera o puede proceder a leer otro registro. Los buffers que contienen registros que ya han sido leídos y no procesados, generalmente se usan para poder mantener varios registros.

El problema principal del buffering es detectar que terminó una E/S tan pronto como sea posible, para poder lanzar la próxima. La solución para este problema son las interrupciones. Cuando un dispositivo de E/S termina con una operación, interrumpe a la CPU. Esta para de hacer lo que este haciendo, y se hace una transferencia de control a la rutina de atención de la interrupción. Esta rutina chequea si el buffer no esta lleno (para un dispositivo de entrada) o vacío (para uno de salida), y lanza el próximo pedido de E/S. Entonces, la CPU resume el cálculo interrumpido. De esta forma, los dispositivos de E/S y la CPU operan a máxima velocidad.

El buffering afecta la performance suavizando las diferencias en las variaciones de tiempo que toma procesar un registro. Si, en promedio, las velocidades de CPU y de dispositivos de E/S son similares, el buffering permite que ambos procesen casi a velocidad máxima.

Sin embargo, si la CPU es mucho más rápida que un dispositivo, el buffering no afecta demasiado la performance, porque la CPU tendrá que esperar que el dispositivo se libere aún cuando todos los buffers disponibles estén llenos.

Esta situación también ocurre con trabajos limitados por E/S (I/O-bound), donde la cantidad de E/S en relación con los cálculos es muy grande. Como la CPU es más rápida que los dispositivos, la velocidad de ejecución esta limitada por la velocidad del dispositivo de E/S. Si, por otro lado, en el caso de trabajos limitados por CPU (CPU bound), donde la cantidad de cálculos es tan grande que los buffers de entrada están siempre llenos, y los de salida vacíos, la CPU no puede emparejarse con los dispositivos de E/S.

11.4.3. Spooling[21][S.O.]

Esta es la mejor solución para todos los problemas mencionados anteriormente.

En muchos sistemas se comenzó a reemplazar los sistemas off-line y a utilizarse discos, al estar estos más disponibles. Los mismos mejoraron la operación off-line, al ser más rápidos que las cintas. El problema con éstas es que no se puede escribir en un extremo de la misma mientras la CPU lee del otro. Los discos eliminan este problema: moviendo la cabeza de un área del disco a la otra, sin el problema del rebobinado, el disco puede cambiar del área de tarjetas que se están usando, para almacenar nuevas tarjetas en otra posición.

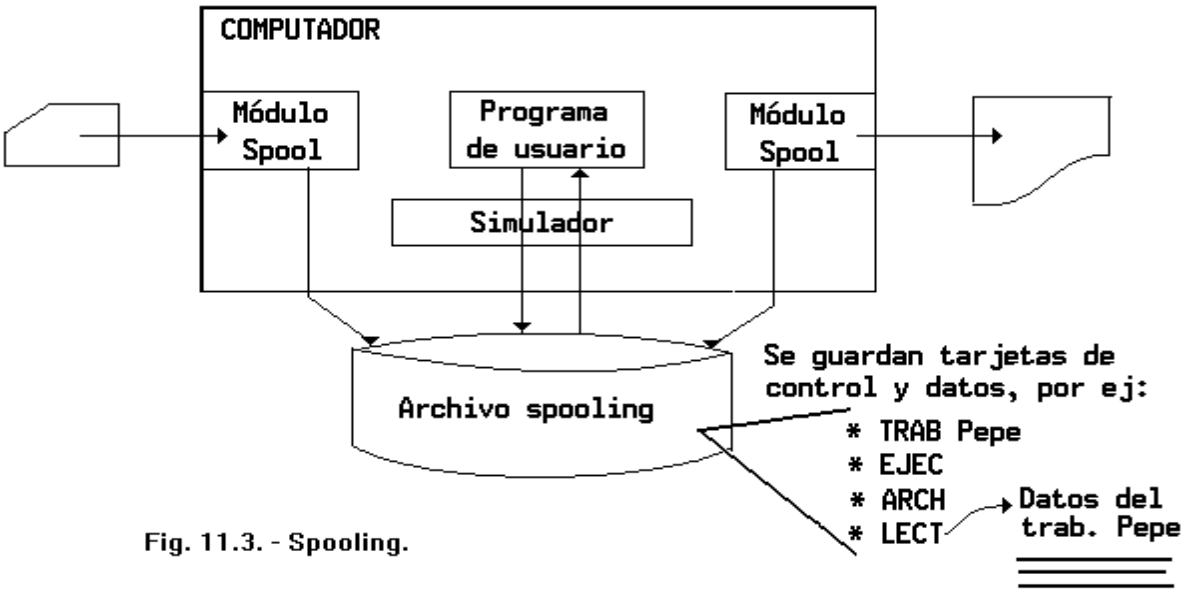

Fig. 11.3. - Spooling.

Basado en esta ventaja se crean los sistemas de SPOOL (Simultaneous Peripheral Operation On-Line). Se leen las tarjetas directamente desde la lectora en el disco. La ubicación de la imagen de las tarjetas en el disco se almacena en una tabla del sistema operativo. Cada trabajo se almacena en la tabla. Cuando se ejecuta un trabajo, sus pedidos para la lectora de tarjetas se satisfacen leyendo su imagen desde el disco. De la misma forma, cuando se pide la impresora, esa línea se copia en el disco. Cuando el trabajo termina, se imprime la salida completa.

El buffering superpone la E/S de un trabajo con sus propios cálculos. La ventaja del spooling es que superpone la E/S de un trabajo con los cálculos de otros trabajos. Esto tiene un efecto directo sobre la performance: la CPU y los dispositivos de E/S se mantienen ocupados con tasas muy altas, particularmente si hay una mezcla de trabajos CPU-bound e I/O-bound.

Además, el spooling provee una estructura de datos importante: una cola de trabajos. El spooling resulta en varios trabajos que han sido leídos y esperan en el disco, listos para ser ejecutados. Esta cola permite al sistema operativo elegir qué trabajo ejecutar luego, para aumentar la utilización de la CPU. Si los trabajos vienen directamente en tarjetas, o aún en una cinta, no es posible ejecutar trabajos en distinto orden al de llegada. En cambio, al tener los trabajos en un dispositivo de acceso directo surge la posibilidad de hacer una Planificación de Trabajos. Esta selección de programas lleva a la multiprogramación, o sea, tener más de un programa cargado en memoria.

11.5. MULTIPROGRAMACION[21][S.O.]

Es un intento de aumentar la utilización de la CPU, logrando que ésta siempre tenga algo que ejecutar. La idea es la siguiente: el SO levanta uno de los trabajos de la cola de trabajos y comienza a ejecutarlo. En el momento en que tenga que esperar por algo (montar una cinta, escribir algo en el teclado, terminar una operación de E/S), la CPU se entrega a otro trabajo, y así sucesivamente. En un sistema monoprogramado, la CPU hubiera estado desocupada, a pesar de que hay trabajo que hacer. Prácticamente por este tema es que surgen todos los administradores a estudiar, y lo que hace (de acuerdo a sus algoritmos), la mayor o menor eficiencia de un SO.

11.6. TIME SHARING[21][S.O.]

Los primeros sistemas batch consisten en el loteo de trabajos similares. Las tarjetas y cintas solo permiten acceso secuencial a los programas y datos, y solo se puede usar una aplicación a la vez.

Cuando aparecen los discos, se puede acceder simultáneamente a todas las aplicaciones. Ahora, los sistemas batch no están más definidos por el loteo de trabajos similares, sino por otras características. *La característica principal de un sistema batch es la falta de interacción entre el usuario y el trabajo mientras está ejecutando. El trabajo se prepara y un tiempo más tarde, aparece la salida.*

El tiempo entre la subsumisión del trabajo y su terminación, llamado el tiempo de turnaround, puede ser el resultado del tiempo de ejecución, o de las demoras antes que el sistema operativo comience a procesar el trabajo.

Como los usuarios no pueden interactuar con sus trabajos mientras están ejecutando, deben colocar tarjetas de control para manejar todos los resultados posibles. En un trabajo multietapa, el resultado puede depender de los resultados anteriores. Puede ser difícil definir que hacer en todos los casos. Otra desventaja es que los programas deben corregirse en forma estática, por medio de dumps de memoria. Un programador no puede modificar un programa a medida que se ejecuta.

Un sistema Interactivo o Hands-on provee comunicación on-line entre el usuario y el sistema. El usuario da instrucciones al sistema operativo o a un programa directamente, y recibe una respuesta inmediatamente, por medio de una terminal. Cuando el sistema operativo termina la ejecución de un comando, pide la próxima "tarjeta de control", no de una lectora de tarjetas sino del teclado. El usuario ejecuta un comando, espera la respuesta, y decide cuál será el próximo comando, basado en el resultado del anterior.

Los sistemas batch son apropiados para ejecutar trabajos grandes, que no tienen interacción. Los trabajos interactivos tienden a estar compuestos por muchas acciones cortas, donde los resultados de un comando puede ser impredecibles. Como el usuario se queda esperando el resultado, el tiempo de respuesta tiene que ser muy corto. Un sistema interactivo esta caracterizado por el deseo de un tiempo de respuesta corto.

Como vimos, los primeros sistemas eran interactivos, pero se perdía mucho tiempo de CPU. Los sistemas de Tiempo Compartido son el resultado de tratar de obtener un sistema interactivo a un costo razonable. Estos sistemas usan multiprogramación y planificación de CPU, para que cada usuario tenga una parte pequeña del tiempo de la computadora. Cada usuario tiene un programa separado en memoria.

Un sistema de tiempo compartido permite a los usuarios compartir simultáneamente la computadora. Como cada acción en un sistema de tiempo compartido tiende a ser corta, solo hace falta un tiempo corto de CPU para cada usuario. Como el sistema cambia rápidamente de un usuario al otro, los usuarios tienen la impresión que cada uno tiene su computadora propia.

11.7. SISTEMAS DE TIEMPO REAL[21][S.O.]

Un sistema de tiempo real generalmente se usa como un dispositivo de control en una aplicación dedicada. Hay sensores que dan datos a la computadora, que los analiza y puede ajustar controles para modificar las entradas del sensor.

Se utilizan para sistemas médicos, controles industriales, controles de experimentos científicos, etc. La característica mas importante de estos sistemas es que tienen restricciones de tiempo bien definidas, y el procesamiento tiene que hacerse dentro de ese tiempo.

11.8. MULTIPROCESAMIENTO[21][S.O.]

Son sistemas con mas de una CPU, compartiendo memoria y periféricos. Se usan dos aproximaciones.

La más común es asignar a cada procesador una tarea específica. Un procesador central controla el sistema, y los demás tienen tareas específicas, o le piden instrucciones al procesador principal. Este esquema define una relación maestro/esclavo. Ejemplos de estos sistemas son aquellos que usan un Procesador Frontal (Front-End Processor) para manejar lectoras e impresoras a alguna distancia del procesador central. Estos sistemas están compuestos generalmente de un computador grande, que es la computadora principal (Host o Mainframe), y un computador más pequeño que es responsable de la E/S de la terminal.

El otro tipo común de multiprocesamiento es el uso de redes. En éstas, varias computadoras independientes pueden comunicarse, intercambiar archivos e información. Cada computador tiene su propio sistema operativo, y opera independientemente.

Actualmente existen en etapa experimental Sistemas Operativos distribuidos, o sea, con características similares a la anterior, pero en donde sus funciones están distribuidas entre distintos procesadores.

11.9. SERVICIOS QUE BRINDAN LOS SISTEMAS OPERATIVOS[21][S.O.]

Un sistema operativo provee ciertos servicios a los programas y a los usuarios. Estos servicios difieren de un sistema a otro, pero hay cierta clase de servicios que pueden identificarse:

- Ejecución de programas: El SO debe ser capaz de cargarlos, darles el control y determinar su fin normal o anormal.
- Operaciones de E/S: El programa del usuario no puede ejecutar operaciones de E/S directamente, por lo tanto el sistema operativo tiene que proveer ciertos medios para hacerlo.
- Manipulación del Sistema de Archivos: Los programas quieren leer y escribir archivos. Esto se hace por medio del sistema de archivos.
- Detección de errores: Estos pueden ocurrir en la CPU y memoria, en dispositivos de E/S, o en el programa del usuario. Por cada tipo de error, el sistema operativo tiene que tomar un tipo de acción distinto.
- Administración de recursos: Cuando hay varios usuarios ejecutando al mismo tiempo, hay que darles recursos a cada uno de ellos. El SO maneja distintos tipos de recursos: CPU, memoria principal, archivos, dispositivos.
- Accounting: Se quiere contabilizar qué recursos son usados por cada usuario. Estos datos, usados con fines estadísticos para configurar el sistema, pueden mejorar los servicios de computación. También se utilizan estos datos para cobrar a cada usuario los recursos utilizados.
- Protección: Los dueños de la información la quieren controlar. Cuando se ejecutan distintos trabajos simultáneamente, uno no debe poder interferir con los otros. Las demandas conflictivas para determinados recursos, deben ser administradas razonablemente.

Los servicios de los sistemas operativos son provistos de distintas formas. Dos métodos básicos son las Llamadas al Sistema y los Programas del Sistema.

El nivel más fundamental de los servicios se maneja por medio del uso de llamadas al supervisor. Estas proveen la interfaz entre un programa en ejecución y el sistema operativo. Generalmente están disponibles como instrucciones del lenguaje ensamblador.

Podemos reconocer, básicamente, los siguientes tipos de llamadas al supervisor:

- Control de Procesos
  - . Fin, Dump.
  - . Carga de otro programa (Load).
  - . Crear procesos, Terminar proceso.
  - . Esperar un tiempo.
  - . Esperar por un evento o una señal.
- Manipulación de Archivos
  - . Crear, borrar archivos.
  - . Apertura y cierre de archivos.
  - . Lectura y escritura.
- Manipulación de Dispositivos

. Pedir un dispositivo, liberar un dispositivo.
. Leer, escribir.

- Mantenimiento de Información
  . Fecha, Hora.
  . Atributos de Procesos, Archivos, o Dispositivos

Hay distintas formas de implementar las llamadas al supervisor dependiendo de la computadora en uso. Puede hacerse como interrupciones o como llamadas a un servidor (server). Es necesario identificar la llamada al sistema que se quiere hacer, y otro tipo de informaciones. Por ejemplo, para leer un registro, hace falta especificar el dispositivo o archivo a usar, y la dirección y longitud de memoria donde copiar la registro.

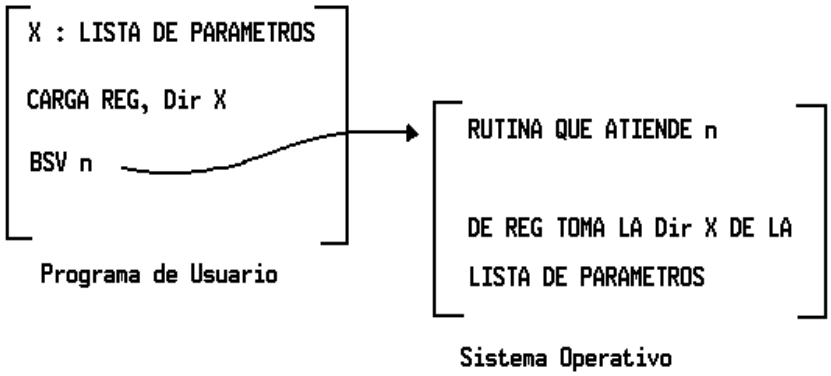

Fig. 11.4. - Pasaje de parámetros al Sistema Operativo.

Hay dos métodos generales para pasar parámetros al sistema operativo. La aproximación más simple es pasar los parámetros en registros. Sin embargo, en algunos casos hay mas parámetros que registros. En estos casos, los parámetros se almacenan en un bloque o una tabla en la memoria, y se pasa la dirección de ese bloque.

Otro aspecto de un sistema moderno es una colección de programas del sistema. Además del código del monitor residente, la mayoría de los sistemas proveen una gran cantidad de programas del sistema para resolver problemas comunes y otorgar un entorno más conveniente para la construcción y ejecución del programa.

Los programas del sistema se dividen en varias categorías:

- Manipulación de archivos: crear, copiar, renombrar, borrar, imprimir, listar, manipular, y hacer vuelcos de archivos y directorios.
- Información de estado: Hay programas para pedir fecha, hora, cantidad de memoria o disco, numero de usuarios, etc.
- Modificación de archivos: editores de texto para crear y modificar los contenidos de archivos.
- Soporte para lenguajes de programación: compiladores, assemblers, intérpretes de los lenguajes de programación más comunes.
- Carga y ejecución de programas: cargadores absolutos, cargadores reubicables, linkeditores y sistemas de debugging.
- Programas de aplicación.

11.10. ESTRUCTURA DE SISTEMAS OPERATIVOS [9][44]

El sistema operativo se divide lógicamente en pequeños módulos y se crea una interfase bien definida para estos módulos.
Cada uno de estos módulos o piezas deben tener su función, sus inputs y outputs cuidadosamente definidos.

El sistema operativos DOS no cuenta con una buena división de estos módulos ya que no se encuentra bien particionado permitiendo el acceso directo de los programas de aplicación a rutinas básicas de E/S para grabar directamente en el display o en los discos, por ello el sistema es vulnerable a estos programas los que pueden provocar el crash del sistema.

La estructura de este sistema puede verse en la figura 11.5.

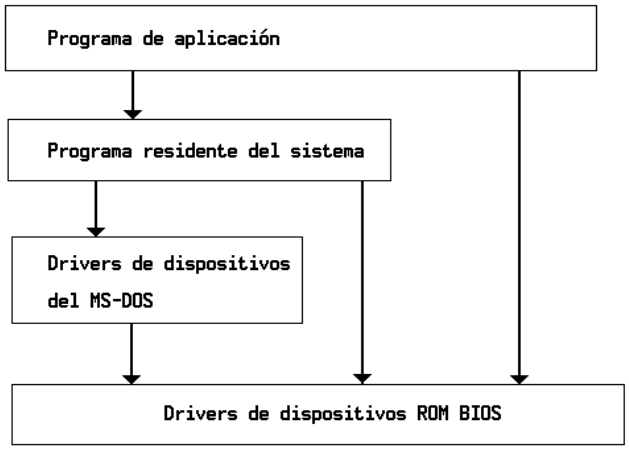

Fig. 11.5.

Otro ejemplo de un sistema operativo que no fue bien construido lo constituye el UNIX. Este sistema se encuentra divido en dos partes una de ellas comprende los programas del sistema y la otra el kernel. El kernel se encuentra divido en drivers de dispositivos y en las interfases (ver figura 11.6).

Lamentablemente este kernel combina demasiada funcionalidad en un solo nivel.

Las llamadas al sistema definen la interfase del programador al UNIX. El conjunto de programas de sistema usualmente disponibles definen la interfase del usuario. Ambas definen el contexto al cual el kernel debe dar

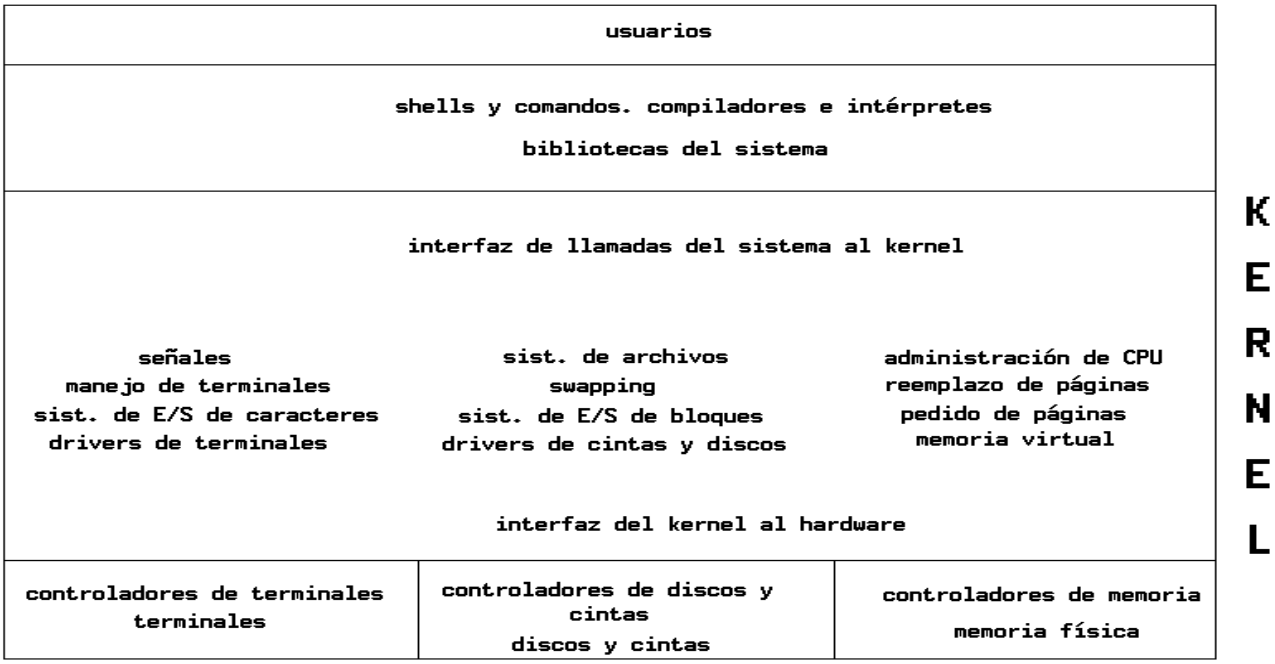

Fig. 11.6.

soporte.

Otras mejoras posteriores al UNIX han separado el kernel en más módulos, por ejemplo en el sistema operativo AIX de IBM se lo dividió en dos partes. El MACH de Carnegie-Mellon redujo el kernel a un conjunto pequeño de funciones básicas trasladando todo lo no esencial a los niveles del sistema o incluso al del usuario.

11.10.1 Diseño en Capas [9][44]

Para mejorar casos como el UNIX se recurre a un diseño en capas, en donde cada capa se construye sobre la anterior. La capa 0 es el hardware y la capa N es la interfase de usuario.

Una capa (por ejemplo la capa M) consiste en algunas estructuras de datos y un conjunto de rutinas que operan sobre esos datos y que pueden ser invocadas por capas o niveles superiores. La capa M puede invocar operaciones en capas inferiores.

La principal ventaja de una estructura en capas es la modularidad. Las capas se seleccionan de tal manera que cada una utiliza funciones (operaciones) y servicios solo de capas de nivel inferior.

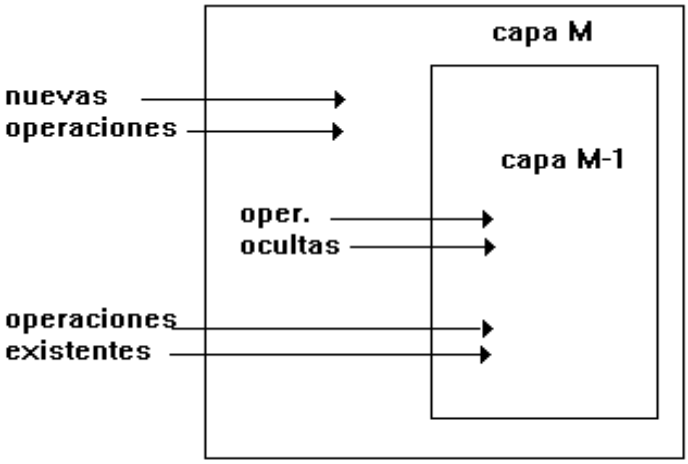

Fig. 11.7. - Un sistema operativo en capas.

Este esquema facilita la depuración aislada de cada capa. El primer nivel puede ser depurado sin tener en cuenta el resto del sistema ya que éste utiliza el hardware básico. Una vez depurado este nivel se puede proceder con el siguiente y así siguiendo.

Una capa no necesita saber cómo se implementan las operaciones de las capas inferiores sino que le basta con saber qué le brindan estas operaciones.

El esquema de capas fue originalmente utilizado en el sistema operativo THE (Technische Hogeschool Eindhoven). Este sistema se definió en 6 capas, de las cuales la capa inferior era el hardware, la siguiente implementaba la administración de la CPU, la siguiente el manejo de la memoria (memoria virtual), la tercera capa contenía los drivers para la consola del operador, y en la cuarta se encuentra el buffering de E/S pensado de esta forma para permitir que los buffers se pudieran utilizar en la memoria virtual y además para que los errores de E/S pudieran ser vistos en la consola del operador.

Por ejemplo, en el sistema VENUS diseñado también en capas los niveles bajos (de 0 a 4) conciernen con las administración de la CPU y la memoria.

La mayor dificultad en el diseño de un sistema en capas radica en definir las capas o niveles ya que un nivel solo puede utilizar niveles inferiores, por lo tanto es necesaria una cuidadosa planificación.

Por ejemplo el driver para manejar los discos de la memoria virtual deberían estar en un nivel inferior a las rutinas de la memoria virtual en sí ya que estas últimas requieren del uso de este manejador.

Problemas de esta índole han provocado un retroceso en el diseño en capas en los últimos años. Se han diseñado menos capas con más funcionalidad proveyendo la mayoría de las ventajas del código modular y evitando los problemas de la definición e interacción de las capas.

El OS/2 un descendiente directo del MS-DOS fue creado para superar las limitaciones de este último. El OS/2 provee multitasking y operación en modo dual como así también otras nuevas mejoras.

En contraposición a la estructura del MS-DOS la estructura del OS/2 se encuentra diseñada en capas que por ejemplo, no permiten el acceso directo del usuario a facilidades de bajo nivel lo que otorga un mayor control al sistema operativo sobre el hardware y un mayor conocimiento de qué recursos está utilizando cada programa de usuario.

En la Figura 11.10 podemos ver la estructura en capas del sistema operativo OS/2.

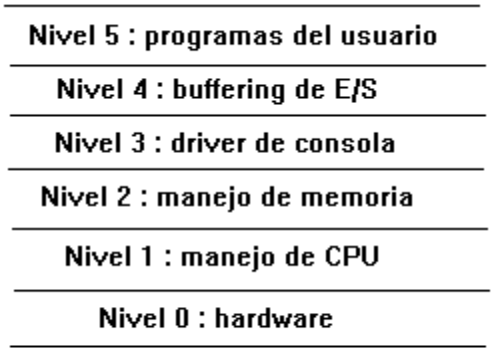

Fig. 11.8. Estructura del THE.

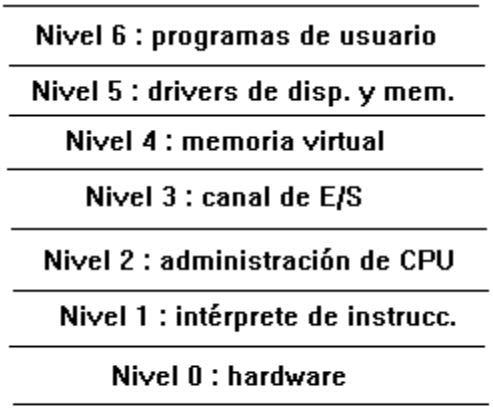

Fig. 11.9. - Estructura del VENUS.

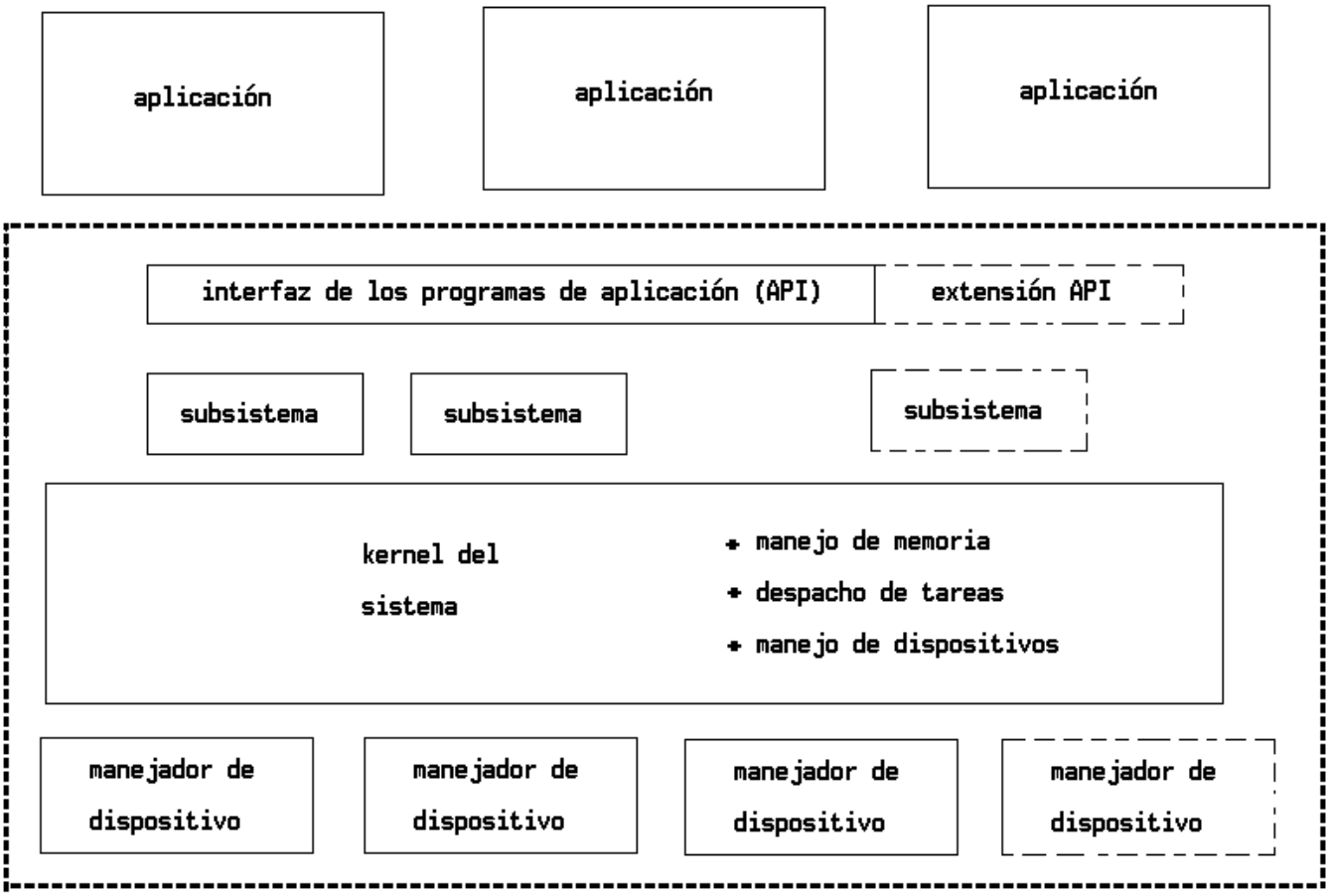

Fig. 11.10. - Estructura en capas del OS/2.

APENDICE I

RESUMEN DE INTERRUPCIONES

Enumeramos a continuación los diferentes tipos de interrupciones existentes, a saber :
- Externas (Consola del operador)
- Fin de E/S.
- Llamadas al SO:
  . Fin (normal o anormal).
  . E/S (en espera, o sin - sincronización -)
  . Información

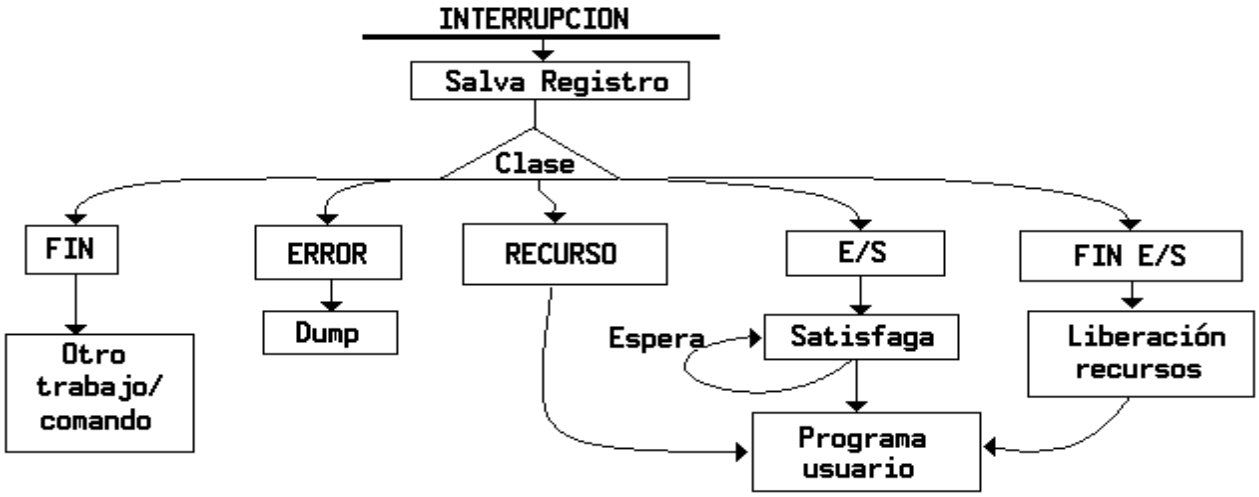

Fig. 11.11. - Diagrama de atención de interrupciones de un solo nivel.

- Errores.
- Reloj de intervalos.

Si tuviésemos un solo nivel de atención de interrupciones, un diagrama sencillo, podría ser el de la Fig. 11.11.

Tener varios niveles de atención de interrupciones da la posibilidad de trabajar en modalidad multitarea.

A lo largo del curso veremos cada uno de los administradores necesarios, y sus distintos tipos y algoritmos posibles para hacer eficiente un SO.

La idea final de la parte de Sistemas Operativos es conocer el flujo de un programa a lo largo de toda su ejecución, sus estados, y las tablas y programas que intervienen en la vida del programa/trabajo/comando.

Damos como adelanto el gráfico de la Figura 11.12.

El esquema general del Sistema Operativo que veremos tendrá una descripción en capas como el de la Fig. 11.13.

Expliquemos brevemente este gráfico:

- Hardware: Set de instrucciones + extracódigos (SVC).
- Núcleo: Implementa Procesos, su comunicación (como semáforos), y administración y sincronización del procesador.
- Administrador de Memoria: Provee servicios de administración de memoria.
- Administrador de Periféricos: Provee servicios de E/S física y administra el uso de los periféricos.
- Administrador de Archivos: Administra el uso y acceso a archivos y provee las abstracciones de registro lógico y archivo.
- Planificador de Trabajos: Administra los recursos de forma global.
- Shell: Maneja la comunicación con el usuario.

Fig. 11.13. - Diseño en capas de un Sistema Operativo.

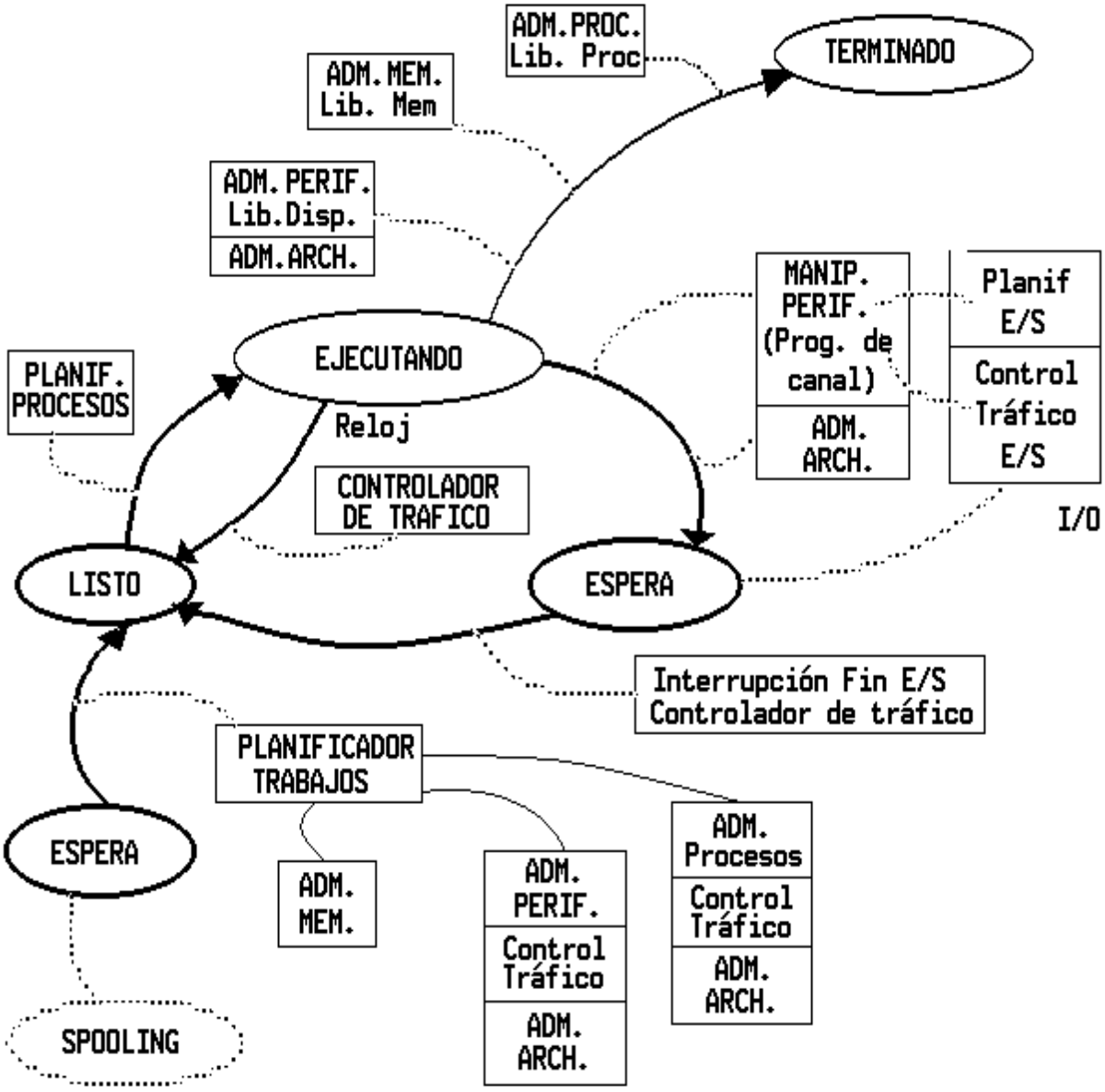

Fig. 11.12. - Un típico Diagrama de Transición de Estados.