CAPÍTULO 2
SOPORTE LÓGICO DE UN COMPUTADOR

En el capítulo anterior distinguimos soporte lógico, de soporte físico a la hora de definir los distintos componentes de un computador. En este capítulo vamos a profundizar y a desarrollar el primero de ellos, cuyo conocimiento será imprescindible para poder proceder al proceso de programación. Muchos de los conceptos desarrollados en este capítulo se deberán confrontar frente al computador donde se valorará su utilidad.

Un mayor desarrollo de los elementos del soporte físico del ordenador están fuera de la óptica de este libro, que trata de enfocar prioritariamente los fundamentos de programación. Sin embargo, existe un gran número de conceptos relacionados con el hardware que no deberían ser ignorados por el estudiante. Por esta razón hemos incluido al final del texto un apéndice que describe los elementos más representativos del hardware actual, a cuyo contenido podrá recurrir el lector si lo considera necesario.

2.1. CONCEPTO DE SOPORTE LÓGICO

El **soporte lógico** o software de un ordenador es el conjunto de programas que permiten realizar las tareas asignadas a la máquina. En este concepto incluimos, tanto los programas suministrados en el momento de adquisición del ordenador, como los adquiridos a empresas de desarrollo y venta de programas y los escritos por los propios usuarios. El soporte lógico, según sea el nivel de trabajo de cada programa, se suele clasificar en **software del sistema** (necesario para administrar y mantener los recursos del ordenador de una forma eficiente) y **software de aplicación** (que corresponde a las aplicaciones específicas que utilizan los recursos del ordenador).

El software del sistema está constituido por:

* **Programa de Arranque**, es el primer programa que se ejecuta cuando arranca la máquina. Comprueba los dispositivos del ordenador y carga en memoria al Sistema Operativo.

* **Sistema Operativo (SO)** conjunto de programas que controlan y supervisan el uso de los recursos del ordenador.

* **Programas de diagnóstico, generación y mantenimiento**. Son utilizados por los responsables del mantenimiento y puesta al día del hardware y del software (incluida la generación y mantenimiento del propio SO). Con estos programas se pretende por ejemplo localizar automáticamente las averías de un determinado dispositivo o circuito, o las causas de un mal funcionamiento de algún modulo del SO.

* **Utilidades generales** y **Herramientas de programación** que contienen programas o ayudas que facilitan la construcción o el uso de las aplicaciones, sea cual sea la naturaleza de éstas. Incluye herramientas tales como:
  - Traductores (ensambladores, compiladores e interpretes).
  - Editores de textos.
  - Rastreadores / depuradores de errores de programación.
  - Gestores de archivos.
  - Administradores de bibliotecas de programas.

Por su parte el software de aplicación es de difícil clasificación, habida cuenta de la diversidad de campos donde se utiliza la informática. Una relación parcial de este tipo de software podría ser:

- Procesadores de texto.
- Bibliotecas matemáticas y estadísticas.
- Hojas de cálculo.
- Sistemas de gestión de archivo y de Bases de Datos.
- Agenda Electrónica.
- Correo Electrónico.
- Aplicaciones Gráficas.
- CAD/CAM (Computer Aided Design/ Manufacturing).
- Gestión de comunicaciones.
- Programas escritos por los usuarios.

Diferenciar si ciertos programas de utilidades son software del sistema o software de aplicación es difícil. Así en principio el SO proporciona todos las características necesarias para el funcionamiento del sistema, sin embargo a veces no alcanzan a cumplir las necesidades del usuario o no es de fácil manejo. Estas deficiencias del SO se cubren mediante las llamadas utilidades, que se desarrollan posteriormente. Algunas de estas utilidades adquieren gran popularidad y las versiones posteriores

del SO las van incorporando. La rápida evolución del software hace que muchos de estos programas de utilidades pasen de ser aplicaciones a ser software básico.

2.2. AYUDAS PARA LA PROGRAMACIÓN

2.2.1 TRADUCTORES

Como vimos en el primer capítulo, un programa escrito en un lenguaje de alto nivel debe ser traducido para que lo pueda ejecutar el computador. Cada traductor de programas es específico de cada tipo de computador y 'entiende' un lenguaje determinado. La transportabilidad de un programa consiste en que un mismo programa escrito en un lenguaje pueda ser entendido por traductores de ese lenguaje para distintos computadores.

No obstante debe tenerse en cuenta que existen muchos "dialectos", siendo necesario con frecuencia adaptar partes de los programas escritos en un dialecto de un lenguaje para pasarlos de una computadora a otra. Existen lenguajes prácticamente "independientes de la computadora" que están normalizados por organismos como el "American National Standars Institute" (ANSI), con el objetivo de garantizar la transportabilidad de los programas (ANSI FORTRAN, ANSI C, etc.).

Para la traducción de los lenguajes de alto nivel, se utilizan los mismos métodos que nosotros usamos cuando tratamos de entender una conferencia expresada en un idioma que desconocemos: una opción es que alguien nos traduzca el contenido completo de la misma una vez concluida, otra es que alguien efectúe una traducción simultánea (frase a frase). Similarmente, según sea su forma de trabajar, existen dos tipos de traductores de programas: los compiladores y los intérpretes.

2.2.1.1 COMPILADORES

Un compilador traduce en su totalidad un programa fuente, escrito en un lenguaje de alto nivel, a un programa objeto, escrito en lenguaje máquina. El programa fuente suele estar contenido en un archivo, y el programa objeto puede almacenarse como otro archivo en un dispositivo de almacenamiento masivo para ser procesado posteriormente, sin necesidad de volver a realizar la traducción.

La traducción por un compilador (la compilación) consta de dos etapas fundamentales: Análisis del programa fuente (tanto lexicográfico como gramatical, llamado también **parsing**) y Síntesis del programa objeto.

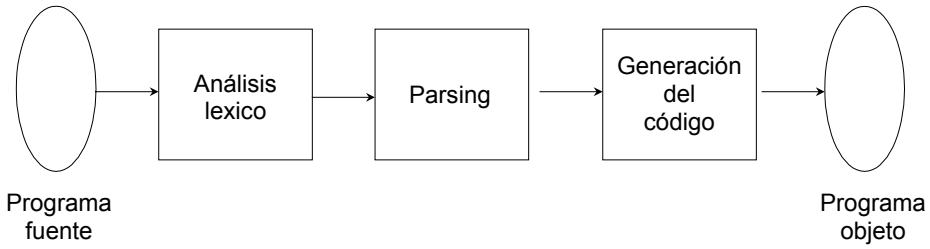

Fig. 2.1. *Etapas de la compilación*

Como todo programa complejo, los compiladores estas diseñados modularmente, de forma que las distintas etapas del proceso de compilación, a su vez, se descomponen en varias fases o módulos, cada una de las cuales puede implicar el recorrer completamente el programa fuente. Vamos a describir brevemente la funcionalidad de cada uno de estos módulos o fases:

a) Análisis lexicográfico.-

Consiste en descomponer el programa fuente en sus elementos constituyentes o símbolos: palabras reservadas, símbolos de operadores, identificadores constantes, comentarios, blancos, etc. Así por ejemplo los símbolos constitutivos de la sentencia:

A=(C+D) * 17

son:
operadores: = ( ) + *
variables: A C D
constantes: 17

b) Análisis sintáctico.

Es la primera parte del análisis gramatical y tiene como misión identificar las estructuras (expresiones, sentencias declarativas, asignaciones, etc.) del programa. La sintaxis de un lenguaje de programación especifica cómo debe escribirse un programa, mediante las reglas correspondientes. Por ejemplo, es sintácticamente incorrecta la sentencia:

A+B= (C-D)* 17

ya que el signo "=" no tiene el sentido que se le da en matemáticas (para formar ecuaciones). En los lenguajes citados dicho signo se utiliza para asignar a una variable (no a una expresión aritmética) el resultado de evaluar una expresión (la que está a la derecha del signo igual). Es correcto sintácticamente escribir:

A=(C-D) * 17

("Calcular (C-D) x17, y el valor que resulte asignárselo a la variable A").

c) Análisis semántico.

La semántica de un lenguaje de programación es el significado dado a las distintas construcciones sintácticas. Por ejemplo, asignar a una variable definida como dato numérico, el valor de una cadena de caracteres, es semánticamente incorrecto para algunos compiladores y si se detecta una circunstancia de este tipo, debe señalarse el error correspondiente.

d) Optimización.

En las fases de optimización se mejora el código intermedio analizándose el programa globalmente. Un programa, por ejemplo (ver Figura 2.2.a) puede incluir dentro de un bucle, que debe ejecutarse diez mil veces, una sentencia que asigna a una variable un valor constante (B= 7,5), no alterándose dicho valor (B) dentro del bucle. Con ello, innecesariamente se asignaría el valor 7,5 a la variable B diez mil veces. El optimizador sacaría la sentencia B= 7,5, fuera (antes) del bucle, ejecutándose así dicha instrucción una sola vez (ver Figura 2.2.b) . Hay que hacer notar que el programa inicial es correcto (con B=7,5 dentro del bucle los resultados del programa son los mismos), pero la optimización realizada por el compilador reduce el tiempo de ejecución.

Desde I = 1 hasta I = 10 000 R= 37.-I*35 B=7,5 Z= B-SIN (-R/35000) fin hacer	B=7.5 Desde I = 1 hasta I=10.000 R=37.-I*35 Z=B-SIN(-R/35000) fin hacer
(a)	(b)

Fig. 2.2. *Ejemplo de optimización de un bucle*

e) Generación de código.-

En esta etapa se genera el código objeto definitivo. El archivo objeto generado puede ser (dependiendo del compilador) directamente ejecutable, o necesitar otros pasos previos a la ejecución, tales como ensamblado, encadenado y carga. Un programa objeto no es todavía ejecutable en la mayoría de los casos ya que es necesario incorporarle las llamadas a funciones estándar, que residen en bibliotecas de programas. Este proceso se llama encadenado (link) y tras él sí se consigue el código ejecutable.

La compilación es un proceso complejo que consume a veces un tiempo muy superior a la propia ejecución del programa. En cualquiera de las fases de análisis, el compilador puede dar mensajes sobre los errores que detecta en el programa

fuente, cancelando en ocasiones, la compilación para que el usuario realice en el archivo fuente las correcciones oportunas.

2.2.1.2 INTÉRPRETES

Una alternativa a la compilación, la cual sólo produce una traducción a lenguaje máquina del programa fuente, es la interpretación que ejecuta el programa directamente desde el programa fuente (Ver Figura 2.3)

Un **intérprete** hace que un programa fuente escrito en un lenguaje vaya, sentencia a sentencia, traduciéndose y ejecutándose directamente por la computadora. El intérprete capta una sentencia fuente, la analiza e interpreta dando lugar a su ejecución inmediata, no creándose, por tanto, un archivo o programa objeto almacenable en memoria masiva para ulteriores ejecuciones.

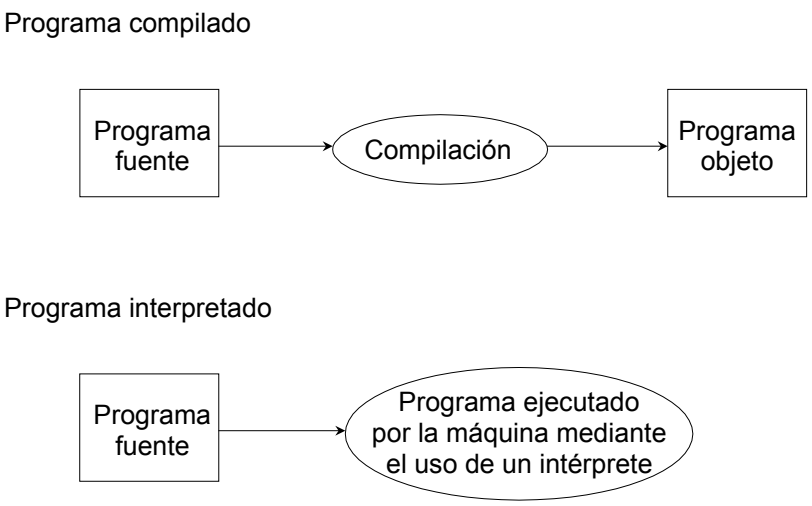

Fig. 2.3. *Compilación e Interpretación*

En la práctica el usuario crea un archivo con el programa fuente (esto suele realizarse con un editor específico del propio intérprete del lenguaje). Según se van almacenando las instrucciones simbólicas, éstas se analizan y se generan los mensajes de error a que dieran lugar; así el usuario puede proceder inmediatamente a su corrección. Una vez creado el archivo fuente, el usuario puede dar la orden de ejecución y el intérprete lo ejecuta línea a línea. El análisis siempre antecede inmediatamente a la ejecución, de forma que:

a) Si una sentencia forma parte de un bucle, se analiza tantas veces como tenga que ejecutarse el bucle. Si el programa de la figura 2.2 a) fuese traducido por un

intérprete, la sentencia R=37.-I*35, se analizaría 10 000 veces, y no sólo una vez, como ocurría en un compilador.

b) Las optimizaciones sólo se hacen dentro del contexto de cada sentencia, y no contemplándose el programa o sus estructuras en conjunto. La optimización mostrada en la figura 2.2 b), no podría ser llevada a cabo por un intérprete, porque cuando llegue a B=7.5 ya se han ejecutado las sentencias anteriores, no pudiendo, por tanto, ubicar la sentencia B=7.5 antes del bucle.

c) Cada vez que utilicemos un programa tenemos que volver a traducirlo, ya que en la traducción no se genera un archivo objeto que poder guardar en memoria masiva (y utilizarlo en cada nueva ejecución). Con un compilador, aunque en muchos casos la traducción sea más lenta, ésta sólo debe hacerse una vez (ya depurado el programa) y cuando deseamos ejecutar un programa ejecutamos el archivo ejecutable creado a partir del objeto.

Los intérpretes, a pesar de los inconvenientes anteriores, a veces son preferibles a los compiladores, como por ejemplo, cuando el número de veces que se va a ejecutar el programa es muy bajo y no hay problemas de velocidad; además, con ellos puede ser más fácil desarrollar programas. Esto es así porque normalmente la ejecución de un programa bajo intérprete puede interrumpirse en cualquier momento para conocer los valores de las distintas variables y la instrucción fuente que acaba de ejecutarse. Cuando esto se hace se tiene una gran ayuda para la localización de errores en el programa. Con un programa objeto esto no se puede realizar, salvo que el programa se ejecute bajo el control de un programa especial de ayuda denominado **depurador** (“debugger”). Además, con un intérprete, cuando se localiza un error sólo debe modificarse este error y volver a ejecutarse a partir de la instrucción en cuestión. Con un compilador, si se localiza un error durante la ejecución, el programador debe corregir las instrucciones erróneas sobre el archivo fuente (no sobre el objeto) y volver a compilarlo en su totalidad después ensamblarlo, y en su caso, montarlo y cargarlo.

Existen en la actualidad traductores que en cierta medida responden de las ventajas tanto de los intérpretes como de los compiladores. Estos traductores se denominan **compiladores interactivos o incrementales**. Un compilador interactivo es un sistema que permite la edición , compilación y ejecución de programas escritos en un determinado lenguaje de alto nivel, simplificando la creación y depuración del programa, permitiendo que el compilador durante la edición realice parte de las comprobaciones y la traducción a código intermedio. Ello permite detectar con antelación muchos errores y evitar la retraducción de todo el programa , si se han realizado sólo pequeños cambios.

2.2.2 TIPOS DE LENGUAJES DE ALTO NIVEL

En los primeros tiempos de la informática y considerando el pequeño tamaño de las memorias disponibles y la lentitud de los procesadores, se pensaba que los traductores iban a ser muy ineficientes; sin embargo con la aparición del primer compilador FORTRAN a mitad de los 50, su uso se extendió rápidamente, hasta que en 1965, se estima que la Babel informática llegaba ya, hasta los 1700 lenguajes de programación diferentes. Esta explosión continua en la actualidad, puesto que la experiencia acumulada y el advenimiento de nuevas tecnologías, tanto en software como en hardware, favorecen la aparición de nuevos lenguajes y el refinamiento y mejora de los existentes. Este frenesí, que conduce a que un lenguaje aprendido hoy, esté destinado a ser declarado mas o menos obsoleto dentro de pocos años, explica que esté fuera del objetivo de este libro, el estudio de un lenguaje particular.

Nuestro objetivo es separar los principios de la informática y la programación, del conocimiento de una sintaxis y una semántica particular. Sin embargo es obvio, que la formación en Informática, no se entiende sin un cierto dominio de al menos un lenguaje de alto nivel(que debe conseguirse delante de un ordenador) aunque la selección del mismo dependerá de las necesidades y posibilidades de cada lector, ó curso.

Con el objetivo de poner orden en esta Babel, en una primera clasificación podemos distinguir entre lenguajes de propósito general (empleados en todo tipo de aplicaciones: de gestión, científico-técnicas, de desarrollo de software de sistemas, etc.) cuyos ejemplos podrían ser el C, Pascal y otros de propósito más específico como sería el caso del FORTRAN para el calculo científico, el Cobol para la gestión o el LISP y el Prolog para la Inteligencia Artificial, por no referirnos a lenguajes específicos para la gestión de bases de datos, como el SQL.

Una segunda clasificación, es posible atendiendo al estilo de programación:

a) Lenguajes basados en la asignación de valores (**Lenguajes Procedurales o Imperativos**). Se fundamentan en la utilización de variables para almacenar valores y en la realización de operaciones con los datos almacenados, especificando la secuencia de acciones que conduce a la resolución del problema propuesto. La mayoría de los lenguajes son de este tipo : FORTRAN, BASIC, COBOL, Pascal, Modula, Ada, C, etc.

b) **Lenguajes declarativos**, que se limitan a describir sus estructuras de datos y las relaciones entre ellas que puedan resultar significativas para la realización de una determinada tarea. Por tanto están basados en la definición de funciones o relaciones y no utilizan instrucciones de asignación , por lo que sus variables

no almacenan necesariamente valores. Los programas están formados bien por una serie de definiciones de funciones (**Lenguajes Funcionales**, como LISP) o de predicados (**Lenguajes de Programación Lógica**, como PROLOG, CLP, etc.).

c) **Lenguajes orientados a objetos**. Estos lenguajes utilizan estructuras de datos complejas, llamados **objetos,** que encapsulan simultáneamente información y operaciones. Estos objetos se comunican entre si a través de mensajes que son reconocibles de forma específica por cada objeto. Un programa es básicamente un universo de objetos que al ejecutarse intercambian mensajes entre sí, simulando sus respectivos comportamientos, en busca del resultado final. Ejemplo de estos lenguajes son el C++, Smalltalk, etc.

2.2.3 UTILIDADES Y FASES EN LA EJECUCIÓN DE UN PROGRAMA

Ya hemos adelantado que una vez codificado un programa, para su ejecución era necesario realizar una serie de operaciones previas. Durante las primeras generaciones de computadoras estas operaciones (o su control) se hacían más o menos manualmente, pero hoy día existen programas de ayuda al usuario, de forma que la propia computadora se usa para auxiliar en la **confección, prueba y control de la ejecución** de los programas. Los programas que proporcionan estas herramientas y ayudas, genéricamente se denominan **utilidades de programación**. Por ejemplo, la ejecución de un programa llamado MEDIAS en un lenguaje de alto nivel, supone seguir el esquema de la Figura 2.4.

Una vez redactado el programa, debe introducirse en la computadora. Para ello crea un archivo en memoria masiva (disco), con ayuda del **editor de textos**. Este es un programa de utilidad que nos permite introducir y modificar (borrar, intercalar, cambiar, ampliar, duplicar, etc.) cómodamente información (de programas o datos) en un archivo. Podríamos decir que esta fase es la introducción y corrección "mecanográfica" del programa.
Cuando el archivo está creado (supongamos que con el nombre de MEDIAS.C en un PC con DOS), pasamos a *compilar* el programa. Con ello obtenemos el mismo programa en lenguaje ensamblador (suponiendo que este compilador no genera directamente código máquina).

FASE	UTILIDAD O AYUDA
1) Introducción y corrección 'mecanográfica' del programa.	Editor de textos ("text editor").
2) Traducción de lenguaje de alto nivel a lenguaje ensamblador.(*).	Compilador ("compiler")
3) Traducción de lenguaje	Ensamblador ("assembler").

ensamblador a lenguaje máquina. (*).	
4) Montaje de un archivo ejecutable, enlazando los módulos que componen el programa completo.	Montador, Enlazador ("linker").
5) Carga del programa en memoria principal.	Cargador ("loader").
6) Ejecución del programa.	Sistema Operativo.
7) Rastreo y depuración.	Rastreador y depurador ("tracer" y "debugger").

(*) En el caso de que el compilador genera el código objeto directamente en lenguaje máquina y no en lenguaje ensamblador estos dos pasos se funden.

Fig. 2.4. *Fases y utilidades en la ejecución de un programa*

A continuación se *ensambla*, generándose un nuevo archivo (en nuestro ejemplo de nombre MEDIAS.OBJ) que contiene el programa en lenguaje máquina. Este programa se suele denominar *reubicable*, y no es directamente ejecutable. En efecto, la mayoría de programas hacen llamadas a diversas rutinas o segmentos del propio usuario o del sistema (funciones de biblioteca, etc.) que deben "unirse" al programa principal. Todos estos segmentos, antes de unirse o **montarse** deben estar compilados y ensamblados.

Todos los segmentos a unir han de ser reubicables, en el sentido de que su direccionamiento es relativo, comenzando la primera instrucción en la dirección 0 de memoria. Además tienen asociados unas tablas donde se encuentran anotadas las direcciones relativas de aquellas instrucciones que hacen referencia a otras direcciones de memoria bien del propio segmento (instrucciones de saltos, por ejemplo) o de otros segmentos (por ejemplo llamadas a subrutina). A veces, el programa principal y sus subrutinas no se cargan consecutivamente, sino que pueden ocupar zonas diversas de la memoria, y en este caso hay que transformar las direcciones relativas de los segmentos en las direcciones físicas en que realmente se ubicarán.

Para efectuar la unión de los distintos segmentos o módulos, "encadenando" o "enlazando" las llamadas entre ellos, se utiliza una denominada *montador, colector, enlazador o encadenador* que genera un nuevo archivo (por ejemplo de nombre MEDIAS.EXE en un PC con DOS) denominado **ejecutable** ya que puede ser ejecutado directamente. El encadenador, para realizar su trabajo, utiliza las tablas de instrucciones con referencia a memoria de los módulos reubicables y genera una **tabla de símbolos externos**, que incluye los nombres y direcciones de las instrucciones a las que hay que saltar desde otros segmentos.

La fase siguiente es introducir o *cargar* el programa ejecutable en memoria y prepararlo para su ejecución. Estas operaciones las realiza una utilidad denominada **cargador**. La siguiente fase es la *ejecución* del programa. Para ello un módulo del sistema operativo pasa el control de la CPU a la dirección de la palabra de memoria donde se encuentra la primera instrucción del programa (es decir, carga en el registro **contador de programa** la dirección física de dicha instrucción) a partir de la cual se sucederá la ejecución de las distintas instrucciones que constituyen el programa.

Las utilidades de *rastreo* y *depuración* de errores son de uso opcional y permiten efectuar funciones tales como ejecutar el programa instrucción a instrucción, mostrándose después de cada ejecución el contenido de las variables que van cambiando; su objeto es el de detectar posibles errores u optimizar el programa.

2.3. PROGRAMA DE ARRANQUE

Cuando se enciende un computador, lo primero que éste hace es llevar a cabo un autodiagnóstico llamado **autoprueba de encendido (Power On Self Test, POST).** Durante el mismo, la computadora identifica su memoria, sus discos, su teclado y cualquier otro dispositivo que tenga conectado. Lo siguiente es buscar un SO para arrancar (**boot**) como primer programa para "empezar a funcionar por sí misma". Ello lleva a preguntarnos en qué momento empieza a trabajar el Sistema Operativo. Para entender este procedimiento hemos de recordar algunos elementos de la tecnología de la memoria principal. Sabemos que la RAM es volátil, mientras que la ROM retiene la información almacenada, aunque se apague el ordenador. En esta última hemos de cargar un software cuyo concurso es imprescindible para la citada identificación del hardware del ordenador y para que se produzca el proceso de arranque de la máquina.

Una CPU está construida de forma que cada vez que se arranca, se inicializa el contador de programa en una dirección predeterminada, antes de su primer ciclo de máquina. Por tanto, de forma automática, interpreta que el contenido de esta posición de memoria, es el inicio del primer programa que va a ser ejecutado. En esta área de memoria ROM se arranca el programa, que entre otras cosas se encargará de cargar el sistema operativo (ver Figura 2.5). Evidentemente este programa en ROM no forma parte del propio sistema operativo, ya que su función consiste, normalmente, en acceder a un disco predeterminado, donde asume que se encuentra el inicio del sistema operativo (**Un PC busca primero el SO en la unidad de disco flexible; si encuentra ahí un SO válido, lo utiliza; si no lo busca en el disco duro primario**) y ejecuta una transferencia del disco al área de memoria donde se instala el sistema operativo (o parte de éste). De esta forma, al arrancarse la máquina, el Sistema Operativo se carga automáticamente en ella, en posiciones que salvo cambios son siempre las mismas (Ver Figura 2.5).

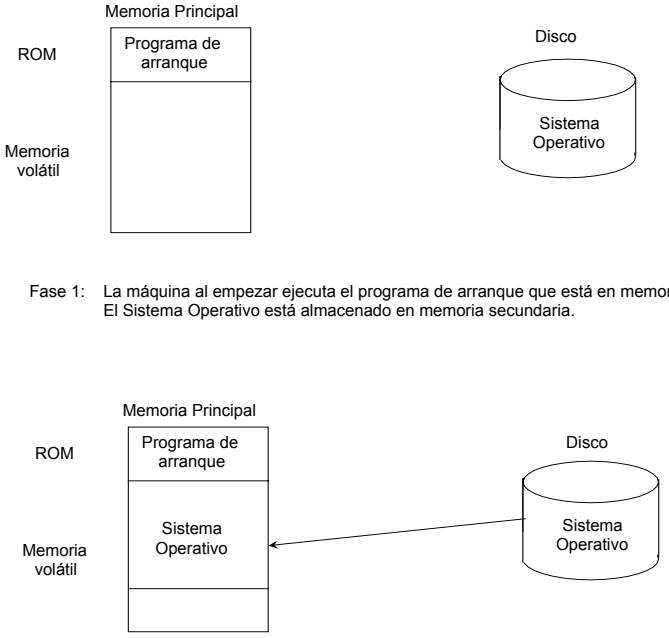

Fig. 2.5. *Proceso de arranque y carga del sistema operativo.*

2.4. SISTEMAS OPERATIVOS (SO)

Un sistema operativo (SO) es en sí mismo un programa de computadora, aunque un tanto especial, quizás el más complejo e importante. El SO es en cierto sentido, una parte integral de la máquina y es tan importante conocerlo como conocer su hardware. Tras el arranque, ya sabemos, que el SO despierta a la computadora y hace que reconozca a la CPU, la memoria, el teclado, y los distintos periféricos. Una vez puesto en marcha el SO, mantiene al menos parte de éste en su memoria en todo momento. Todos los SO tienen dos objetivos fundamentales: a) Hacer posible el uso eficiente de los recursos del sistema y b) Ocultar las dificultades que supone el control directo del hardware del ordenador. El primer objetivo se ve entorpecido por el hecho de que algunos dispositivos funcionan mucho más rápidamente que otros; por ello, una de las misiones de los SO es la de garantizar que los dispositivos más rápidos, como los procesadores, no sean retenidos por otros dispositivos más lentos, como los dispositivos periféricos. Para conseguir el segundo, simplificar las operaciones del hardware, los SO generan una **máquina virtual**. Una máquina virtual es un ordenador simplificado, en el que los

detalles corren a cargo del SO. Las personas que escriben programas sólo necesitan conocer la máquina virtual, sin necesidad de entrar en los detalles reales del hardware. Una consecuencia importante de ello es que una aplicación desarrollada para un SO, puede que sólo necesite recompilarse, para pasar de un entorno hardware a otro.

2.4.1 FUNCIONES DE LOS SISTEMAS OPERATIVOS.

Las tareas que desempeña un SO dependen en cierta medida del tipo de las características de cada ordenador. Así las funciones de un SO multitarea, en un gran ordenador, son diferentes de las de un ordenador personal. Sin embargo, existen ciertos puntos en común que permiten clarificar de forma general las tareas principales de un SO en:

- Proporcionar una **interfaz de usuario**, para que este último se pueda comunicar con la computadora.
- Administrar y controlar los dispositivos de hardware del computador.
- Administrar y mantener los sistemas de archivo de disco.
- Apoyar la ejecución de otros programas.

En los grandes sistemas, existe un **operador** (o un equipo de ellos) cuya misión está relacionada con el SO, ellos arrancan y paran el ordenador, cargan programas y discos de datos, asignan recursos a los usuarios y responden a cualquier problema que pueda presentarse. Las redes de ordenadores suelen tener, además, un administrador de red cuyo trabajo consiste en mantener la red funcionando de forma eficiente, principalmente controlando los recursos compartidos como servidores de ficheros e impresoras. Estos operadores de ordenadores y administradores de redes trabajan directamente con el SO de los ordenadores. En los PC y estaciones de trabajo el usuario es también el operador, activando y parando programas, cargando disquetes, etc.

Los SO como programas deben tener una serie de características relacionadas con su calidad, en particular: **eficiencia**, (supone que el SO debe ejecutar sus funciones de forma rápida ya que el tiempo que el SO emplea en su funcionamiento es tiempo no disponible para la aplicación que se ejecuta), **fiabilidad**, (un fallo en el SO puede inutilizar el ordenador que éste controla), facilidad de **mantenimiento** y un **tamaño reducido** (un SO pequeño ocupa menos espacio es menos propenso a los errores y funciona más rápidamente). En la práctica es necesario llegar a un equilibrio entre las facilidades que proporciona el SO y su tamaño.

2.4.2 LA ESTRUCTURA DE UN SISTEMA OPERATIVO TÍPICO

La estructura de un SO varía de acuerdo con su tamaño y complejidad. Sin embargo, las partes o módulos que vamos a citar se encuentran en todos los SO de una u otra forma. Los encargados de desarrollar sistemas operativos utilizan la metáfora de una semilla para describir la estructura de los programas que escriben. Así las funciones centrales de un SO son controladas por el **núcleo (kernel)** mientras que la interfaz del usuario es controlada por el **entorno (shell**). Por ejemplo, la parte más importante del DOS, es un programa con el nombre COMMAND.COM. Este programa tiene las citadas dos partes: a) El núcleo, que se mantiene en memoria en todo momento contiene el código máquina de bajo nivel para manejar la administración del hardware que pueda necesitar el programa que se está ejecutando. b) La shell, que también se le llama intérprete de órdenes (es quien toma el control de la pantalla), permite que el usuario teclee, interpreta lo tecleado y lo lleva a cabo. El intérprete de órdenes es la parte del programa que establece la interfaz de línea de órdenes.

En los sistemas DOS, al menos una parte del programa COMMAND.COM está siempre en la memoria, proporcionando a los programas los servicios de bajo nivel administración del hardware y del disco. Sin embargo, las funciones de bajo nivel del SO y las funciones de interpretación de órdenes están separadas, de tal forma que es posible reemplazar el intérprete de órdenes MS-DOS estándar con uno diferente. En otras palabras, puedes mantener el núcleo del DOS ejecutándose, pero utilizar una interfaz de usuario diferente. Esto es exactamente lo que sucede cuando cargas Windows. Al teclear "WIN" en el prompt de DOS, el programa Windows toma el lugar de la shell, reemplazando la interfaz de línea de órdenes con una interfaz gráfica del usuario.

2.4.2.1 NÚCLEO

El propio núcleo es el módulo de más bajo nivel del SO y trabaja al servicio del resto de módulos. El núcleo trabaja directamente sobre el hardware del ordenador y proporciona una serie de servicios a las capas superiores del sistema. Entre estas tareas se incluyen:

- Manejo de interrupciones: Entendemos como **interrupción** toda señal externa que detiene la ejecución de un programa. Cuando el hardware del ordenador detecta una interrupción, el control se transfiere al módulo de control de interrupciones del núcleo, que analiza el carácter de la interrupción y toma las acciones apropiadas (transferir el control a otro módulo del SO, iniciar otro programa o continuar la ejecución del programa interrumpido, etc.).

- Asignación de trabajo al procesador: Para ello el núcleo transfiere el control al programa que el planificador ha determinado, para que sea el próximo en ejecutarse. El control se logra manteniendo una cola de mensajes en espera para cada uno de los programas activos. El núcleo recibe los mensajes y los va almacenando en la cola apropiada al destino en cuestión, para distribuirlos cuando el programa de destino se active.

- Comunicación entre programas: La mayoría de los ordenadores disponen de instrucciones cuyo uso está restringido al núcleo, que transfieren el control de un programa a otro y que acceden a determinados registros.

2.4.2.2 MÓDULOS DE INTERFAZ CON EL OPERADOR Y LOS USUARIOS

Las comunicaciones entre el SO y el mundo exterior tienen lugar a través de dos interfaces: el interfaz usuario/SO y el interfaz SO/operador. Esta idea puede resultar extraña para el usuario habitual de un ordenador personal, pues en estos sistemas, se confunden en uno solo. El interfaz con el usuario proporciona un lenguaje, cuyas instrucciones controlan la ejecución de cada programa, especificándose entre otras cosas el máximo tiempo de CPU que se puede utilizar, cuánta memoria se necesita y qué periféricos se van a utilizar. El interfaz SO/operador del sistema está constituido también por órdenes y mensajes de forma que el operador puede dirigir gran parte de las funciones del SO. Esto se aplica particularmente a la planificación y asignación de recursos. En todo momento el operador debe tener el control global del ordenador.

Existen dos amplias categorías de interfaz de usuario: interfaces de línea de órdenes o interfaces gráficas de usuario. Para usar un SO con **interfaz de línea de orden**, se introduce palabras y símbolos desde el teclado de la computadora. Con un interfaz **gráfico del usuario (Graphical User Interface, GUI**), se seleccionan las acciones mediante el uso de un ratón o dispositivo indicador similar para pulsar sobre figuras llamadas iconos o seleccionar opciones de los menús. Cada SO proporciona una interfaz de usuario, de cualquiera de los tipos ya descritos. (Ya hemos visto que esto lo hace Windows respecto a DOS).

La interfaz de línea de ordenes.-

Como ejemplo, la interfaz de línea de órdenes permite controlar las funciones mediante el tecleo de órdenes, después del indicador de petición de entrada o prompt. En DOS, el prompt por omisión es la letra que identifica la unidad activa de disco seguida de un signo mayor que (C>). El prompt indica que el SO está listo para aceptar una orden. Para introducirla, se utiliza el teclado para teclear las palabras y los símbolos. Si se teclea una orden en forma incorrecta, el

SO responde con un mensaje indicando que no entendió la orden. Cuando esto pasa, simplemente se vuelve a teclear la orden correctamente.

La interfaz gráfica del usuario.-

Mucha gente piensa que el desarrollo más significativo en el mundo de las computadoras desde que los fabricantes comenzaron a construirlas en torno a microprocesadores, fue el desarrollo de la interfaz gráfica del usuario. Al fin, las computadoras permitían trabajar de la misma forma en que la gente trabaja: visualmente. Las interfaces más intuitivas utilizan objetos y símbolos llamadas **iconos** con los cuales cualquier persona está familiarizada, incluyendo aquellas personas que nunca antes han utilizado una computadora. Por ejemplo, todo el mundo sabe lo que es un bote de basura. Es bastante obvio para lo que sirve un icono representando un bote de basura: para tirar objetos. Tú te deshaces de algo llevando su icono al bote de basura. Cuando sueltas el botón del ratón, el objeto desaparece en el bote de basura, que se abulta.

2.4.3 ADMINISTRACIÓN DEL HARDWARE

Sin importar qué tipo de interfaz del usuario tenga la computadora (de línea de ordenes o gráfica), el SO intercepta las órdenes para usar la memoria y otros dispositivos, mantiene un registro de qué programas tienen acceso a qué dispositivos, y así sucesivamente. Así por ejemplo, cuando se introduce la orden directorio (dir) en el prompt del SO o se hace clic en una carpeta de una GUI, el SO interpreta la acción como una orden para mostrar los archivos en ese directorio o carpeta. La lógica del programa en el núcleo responde a la orden mediante la interrupción de la CPU y la instruye para que vaya a la unidad de disco y muestre los nombres de los archivos que encuentre en el directorio o en la carpeta. El SO intercepta la cadena de datos (los nombres de los archivos) regresándolos del disco y desplegándolos en la pantalla. Veamos con un poco mas de detalle, la interacción del SO con los distintos elementos del hardware del ordenador:

2.4.3.1 GESTIÓN DE LA MEMORIA

La memoria principal de la mayoría de los ordenadores es mucho más pequeña de lo que sería necesario para contener todos los programas y datos que maneja un ordenador en un momento dado. El módulo de gestión de la memoria de un SO es el encargado de asignar ciertas porciones de la memoria principal a los diferentes programas o partes de los programas que la puedan necesitar, mientras el resto de los datos y los programas se mantienen en los dispositivos de almacenamiento masivo. De este modo, cuando se asigna una parte de la memoria principal se hace de una forma estructurada, siguiendo una determinada orden.

La forma más común de gestión de la memoria supone crear una **memoria virtual** utilizando los dispositivos de almacenamiento masivo como si fueran parte de la memoria principal. El SO se encarga de controlar las transferencias entre los medios de almacenamiento masivo y la memoria, creando la sensación de una memoria mayor de la que en realidad existe. De este modo, para un usuario la memoria principal y la memoria de almacenamiento masivo forman parte de la misma cosa, la memoria virtual del sistema.

2.4.3.2 CONTROL DE ENTRADA/SALIDA

Los problemas asociados con la E/S de datos tienen su origen en las diferentes características y velocidades de los dispositivos. Por ejemplo, una impresora de líneas produce una línea de caracteres cada vez, mientras que un teclado acepta un único carácter cada vez. Una impresora de líneas tiene una velocidad de transferencia de caracteres (más de cien veces mayor) que un teclado.

El módulo de control de E/S de un SO trata estos problemas presentando al programador la entrada/salida como una cuestión independiente del dispositivo. Para los programadores, todos los dispositivos tienen las mismas características, siendo el SO el encargado de atender las particularidades de cada uno de ellos. Puesto que el sistema operativo es único, se entenderá que los detalles de cada periférico del sistema deban residir en ROM, para que esta información que maneja el SO no se pierda cada vez que se apague el ordenador. Además, también existen "**drivers**" (conductores) que son programas que se ejecutan en cada arranque para hacer de interfaz entre el SO y el periférico.

2.4.3.3 GESTIÓN DE LOS DISPOSITIVOS DE ALMACENAMIENTO MASIVO

Los dispositivos de almacenamiento masivo de un ordenador constituyen la zona donde se guardan datos y programas tanto del sistema como de los diferentes usuarios. El módulo encargado de la gestión de estos dispositivos tiene la misión de mantener la estructura de esta información y de asegurar el uso eficiente de los medios de almacenamiento masivo. El SO se encarga de los aspectos físicos de la transferencia (determinar qué bloques y sectores de un disco se van a utilizar, etc.), dejando al programador libre para que sólo se preocupe de los aspectos lógicos de la transferencia (los registros y ficheros involucrados).

Los datos y los programas de un dispositivo de almacenamiento masivo se mantienen en ficheros. Como veremos a continuación, el módulo de gestión de estos dispositivos de almacenamiento masivo supervisa la creación, actualización y

eliminación de los ficheros que existen en el sistema en cada momento y coopera con el módulo de gestión de la memoria durante las transferencias de datos desde y hacia la memoria principal. Si se dispone de un sistema de memoria virtual, existen transferencias entre la memoria principal y los medios de almacenamiento masivo para mantener la estructura de la memoria virtual.

2.4.4 ADMINISTRACIÓN DEL SISTEMA DE ARCHIVOS

Sabemos cómo un controlador de disco y una unidad de disco trabajaban juntos para almacenar bits y bytes de información en un disco, pero no consideramos la información que el SO pasa a la computadora para leer y almacenar *archivos*. Los archivos pueden contener instrucciones de programas o información creada o usada por un programa. Cuando hay cientos de archivos en un disco, encontrar lo que se necesita puede tomar su tiempo, para reducirlo, se necesita usar los medios proporcionados por el SO para organizar estos archivos dentro de grupos más pequeños y más lógicos, habitualmente en forma de directorios.

Los ficheros almacenados en los dispositivos de almacenamiento masivo contienen información que puede ser compartida, de carácter privado, o incluso secreta. Por tanto, cada fichero está dotado de un conjunto de privilegios de acceso, siendo una misión del SO asegurar que estos privilegios no sean violados.

2.4.5 APOYO A LA EJECUCIÓN DE PROGRAMAS DE APLICACIÓN

Otra de las funciones importantes del SO es proporcionar servicios a otros programas. A menudo, estos servicios son similares a aquellos que el SO proporciona directamente a los usuarios. Por ejemplo, cuando se quiere que un procesador de texto recupere un documento, con el cual se ha estado trabajando, el procesador desplegará los archivos del directorio que se especifique. Para hacer esto, el programa llama al SO para que muestre los archivos. El SO lleva a cabo el mismo proceso para construir una lista de archivos sin importar si la solicitud viene directamente del usuario o de un programa de aplicación, (cuando la solicitud viene de una aplicación, el SO envía los resultados de su trabajo al programa de aplicación en lugar de mandarlos directamente a la pantalla de la computadora).

Algunos otros servicios que el SO proporciona a los programas es guardar archivos en el disco, leerlos de disco a memoria, revisar espacio disponible en disco y en memoria, ubicar memoria para guardar información de un programa, leer la captura en el teclado y desplegar caracteres o gráficos en la pantalla. Cuando los programadores escriben programas de computadora, incluyen en sus programas instrucciones que solicitan los servicios del SO. Estas instrucciones son conocidas

como **llamadas del sistema** debido a que el programa tiene que llamar al SO, para conseguir alguna información o servicios.

La mayor parte del tiempo en que un ordenador está funcionando, la demanda de recursos es mayor que los que realmente existen. Para resolver este problema los SO disponen de una política de administración, a lo largo del tiempo y según las demandas, de estos recursos. El **planificador** es el responsable de llevar a la práctica esta política. El mecanismo sería muy sencillo si fuera posible utilizar una política directa, del tipo "se atenderá primero al que antes lo solicite". Esta política puede llevar a situaciones de **bloqueo**. Esto sucede cuando dos programas quedan bloqueados al solicitar insistentemente recursos que están asignados al otro, como ocurre cuando dos vehículos pesados intentan atravesar simultáneamente un puente estrecho. El planificador se ocupa fundamentalmente de asignar tiempo del procesador a los programas de acuerdo a una cierta política, que varía notablemente de un SO a otro.

Cuando distintos programas se ejecutan simultáneamente, es necesario protegerlos entre sí, incluido del propio SO. Esta protección debe ocurrir especialmente, frente a errores y al abuso deliberado de los recursos del sistema. Aunque es imposible para el SO prever los errores de los programas de aplicación, es esencial detectarlos y diagnosticarlos lo antes posible para limitar sus efectos. Un abuso deliberado del sistema es más difícil de resolver.

La protección de la memoria principal es el aspecto más importante de la seguridad de un ordenador, ya que interviene en todos los procesos. El módulo de gestión de la memoria asigna a cada tarea una parte de la memoria principal del ordenador y a continuación asigna a cada una de estas porciones un grado de protección según la naturaleza de las tareas que tiene encomendadas. Se realizan comprobaciones reiteradas para asegurar que no se producen violaciones de la memoria. La forma más común de protección contra la violación de memoria se da cuando un programa trata de leer o escribir datos fuera del área de memoria que tiene asignada.

2.4.6 MÓDULOS PARA LA GESTIÓN DE REDES

Después de haber indicado la importancia de las redes de computadores, no podemos terminar esta descripción de las funciones de un SO, sin indicar las nuevas demandas a las que tienen que hacer frente éstos, para gestionar redes de ordenadores. La primera consideración a tener en cuenta, es la diversidad de topologías que se pueden dar en una red (en anillo, en bus, irregular, etc.), a las cuales tienen que adaptarse cada uno de los SO que gestiona cada una de las máquinas (Ver sección 2.5). Para que una red pueda operar, los protocolos deben ser gestionados debidamente y cada uno de estos SO debe estar en condiciones de

trabajar con el soporte lógico de las redes de computadores (que trataremos con más detalle en un apartado posterior de éste mismo capítulo). Otras funciones que en este escenario de red debe desarrollar el SO son: controlar la seguridad de las transacciones y evitar accesos no autorizados o vandálicos (introducción de virus en la red, bloqueo intencionado de la misma, etc.)

2.4.7 EJEMPLOS DE SISTEMAS OPERATIVOS

2.4.7.1 MS DOS

El MSDOS (MicroSoft Disk Operating System) es un SO, diseñado inicialmente para un único usuario para los ordenadores que utilizan la familia de microprocesadores Intel 8086 -286-386-486 y Pentium. Fue desarrollado inicialmente en 1979 por Tim Paterson, que trabajaba en Seattle Computer Products. Siendo adquirido y terminado por Microsoft Corporation (con motivo de la aparición del IBM-PC) que es quien lo distribuye y ha realizado sucesivas extensiones, con el objetivo de adaptarlo a las capacidades de los nuevos ordenadores.

CARACTERÍSTICAS GENERALES

El espíritu del MSDOS es el de proporcionar una base para el *software* de un sistema, capaz de controlar todos los aspectos de la operación de un PC, particularmente el sistema de gestión de ficheros en disco, la transferencia de datos entre periféricos y la carga y ejecución de programas. El MSDOS dispone de un procesador de órdenes que llama al sistema de entrada/salida (E/S) y a las utilidades del sistema. El sistema de (E/S) tiene tres niveles: el sistema de gestión de ficheros, el sistema básico de (E/S) (BIOS) y las rutinas *firmware* de E/S, mientras que las operaciones a nivel de detalle, especialmente las transformaciones entre estructuras físicas y lógicas, corren a cargo del BIOS. El sistema de E/S dispone de dos formas de comunicación, una para las unidades de disco, que constituye el sistema de gestión de ficheros, hacia las que se transfieren los datos en bloques y otra para los periféricos hacia los que se transfieren los datos carácter a carácter.

PROCESADOR DE ÓRDENES

El procesador de órdenes tiene cuatro funciones: actuar de interfaz con el usuario, gestionar el sistema de interrupciones, tratar los errores y ejecutar las órdenes internas del MSDOS. Este procesador transfiere el control al sistema de E/S, a una de las utilidades del MSDOS o, a través del sistema de gestión de ficheros, a un programa, según los casos.

El interfaz con el usuario es un conjunto de mensajes emitidos por el ordenador bien en respuesta a órdenes del usuario, bien de forma autónoma (prompt). El MSDOS dispone de facilidades de edición que permiten a los usuarios corregir las órdenes a medida que los van tecleando. También dispone de un editor de líneas para crear ficheros batch, los cuales contienen varias órdenes del propio SO que se van ejecutando secuencialmente una a una al llamar al fichero *batch*. Como ya hemos dicho todas estas características se han visto superadas con el advenimiento de las distintas versiones de Windows.

El sistema de interrupciones dispone de una jerarquía sencilla de prioridades para tratar las interrupciones ocasionadas por los periféricos. Cuando el tratamiento de una interrupción termina, se devuelve el control al programa que se estaba ejecutando cuando sucedió la interrupción. El sistema de tratamiento de errores funciona de una forma muy similar, devolviendo el control al programa en que se produjo el error si ello es posible, y si no al MSDOS.

Varias órdenes del MSDOS dependen directamente del procesador de órdenes. Estas son las órdenes que permiten conocer el directorio de un disco, borrar, cambiar el nombre y copiar ficheros. El MSDOS mantiene un control de la hora y el día, pudiéndose ajustar ambos mediante órdenes.

GESTIÓN DEL SISTEMA DE ALMACENAMIENTO MASIVO

La tarea más importante que desarrolla el MSDOS es la de controlar el sistema de gestión de ficheros del ordenador. Cada disco dispone de un directorio, que contiene los detalles de todos los ficheros del disco, así como los nombres de los subdirectorios. De esta forma los directorios constituyen una estructura jerárquica, en forma de árbol.

En cualquier momento el usuario está “conectado” a un determinado directorio (el directorio por defecto), y , a menos que se especifique otra cosa, todos los ficheros se buscan o crean en ese directorio. Los nombres de los ficheros están compuestos por un identificativo seguido de una extensión de fichero de tres letras, que sirve para agrupar a los ficheros de un mismo tipo. Para localizar un fichero fuera del directorio por defecto hay que utilizar un *pathname*, que contiene los nombres de todos los directorios por los que hay que pasar para llegar al fichero en cuestión. Por ejemplo, el pathname que conduce a un fichero de nombre PBA850.TXT (en este caso la extensión sería TXT), situado en el subdirectorio CARTAS del directorio ADMIN del disco B, sería:

B:\ADMIN\CARTAS\PBA850.TXT

El sistema de gestión de ficheros controla esta estructura jerárquica. Dispone de facilidades para crear un fichero, abrir uno para lectura o escritura, transferir

información entre dos ficheros en el sentido que se indique y para informar del espacio disponible. Las transferencias tienen lugar sector a sector (un sector está compuesto por un bloque de 512 bytes), pudiéndose acceder a los registros de los ficheros de forma secuencial o aleatoria. El sistema de gestión de ficheros se encarga de los detalles relativos a la carga y ejecución de los programas de usuario y trata la mayoría de las llamadas que éstos hacen al SO.

El espacio en disco se asigna tomando el siguiente sector disponible. Cuando un fichero se borra o se reduce su longitud, se liberan los sectores que no se necesitan. No existe un mecanismo de agrupación, y algunos ficheros, en particular, los que se usan para tratamiento de textos y que cambian cada vez que se editan, pueden fragmentarse notablemente. Si un fichero está muy disperso entre los sectores del disco, el acceso al fichero para lectura o escritura puede ser muy lento.

ENTRADA/SALIDA ORIENTADA AL CARÁCTER

Con la excepción de las unidades de disco, todos los datos se envían a todos los dispositivos periféricos carácter a carácter. Estos incluyen el teclado, la pantalla, la impresora y las líneas de comunicaciones. El MSDOS dispone de un conjunto de controladores de dispositivos, uno para cada uno de los dispositivos anteriores, que presentan un interfaz estándar para el programador. De esta forma todos los dispositivos parecen tener las mismas características a los ojos del programador.

UTILIDADES

Las utilidades del MSDOS, van creciendo a medida que aparecen nuevas versiones. Entre las utilidades existentes, se incluyen: formatear, copiar y verificar el estado de los discos, recuperar ficheros perdidos, conexión a red, trabajo en grupo, etc. Además se cuenta con: un editor de líneas, un depurador para detectar errores en los programas en ensamblador y un montador/enlazador para generar módulos ejecutables a partir de código relocalizable. Existe también un programa de ordenación para disponer las líneas de un fichero de texto en orden alfabético.

NUEVAS CARACTERÍSTICAS

La limitaciones del MSDOS, frente a la creciente potencia de los nuevos PC, han propiciado la aparición de aplicaciones tales como Windows que corren bajo MSDOS que permite a los usuarios interactuar con varios programas a la vez. Este *software*, crea una ventana en la pantalla para cada programa que está activo. Utilizando un ratón el usuario mueve el puntero a una ventana particular para interactuar con el programa que se está ejecutando en ella. A su vez las ventanas pueden disponerse en la pantalla y dimensionarse en la forma en que se desee. Los programas que se ejecutan bajo MS Windows disponen de interfaces de usuario

estándar y utilizan menús desplegables. Los datos pueden transferirse de un programa a otro, por ejemplo, una porción de una hoja de cálculo puede incluirse en un documento de un procesador de texto.

La evolución de estas aplicaciones está dando lugar a nuevo sistemas operativos, de tipo gráfico (Windows NT, Windows 95) totalmente diferentes de MSDOS. Estos nuevos sistemas operativos tratan de superar las carencias derivadas de los orígenes del MSDOS que impiden aprovechar todo el potencial de los nuevos procesadores.

2.4.7.2 UNIX

El SO UNIX fue diseñado en los Bell Laboratories de la American Telephone and Telegraph Corporation (AT&T) que todavía siguen dándole soporte. La primera versión entró en funcionamiento en 1971. El UNIX se ha actualizado en varias ocasiones desde entonces y en este proceso de modificación se ha adaptado para usarse en una gran variedad de máquinas. Su implementación en el Cray-2 fue un espaldarazo notable, pues se demostró que podía ser un SO apropiado para un gran rango de sistemas, desde el PC hasta el superordenador. El UNIX combina las características de simplicidad, facilidad de uso y potencia con un pequeño tamaño y una considerable flexibilidad. En un SO con un núcleo pequeño que cuenta con un amplio grado de aceptación, especialmente entre los usuarios experimentados. Aunque tiene ya varios años, su popularidad sigue en aumento. La característica más importante del UNIX es que se ha convertido en un estándar, proporcionando una plataforma estándar, para desarrollar un gran número de aplicaciones y asegura, en la medida de lo posible, la transportabilidad del software entre sistemas UNIX sobre distinto hardware.

CARACTERÍSTICAS GENERALES

El UNIX es un SO de propósito general, multiusuario y multiproceso. Soporta multiprogramación y multiacceso. Dispone de un ensamblador, varios compiladores para lenguajes de alto nivel y de un editor de texto, entre otros elementos. El UNIX está pensado para que varios usuarios puedan acceder a un único procesador a través de terminales. Ningún usuario recibe la consideración de operador del ordenador. Todos los usuarios pueden enviar órdenes al SO durante su trabajo y el SO va respondiendo a estas órdenes.
El UNIX implementa el concepto de memoria virtual, cada usuario parece tener acceso a la totalidad de la memoria del ordenador.

CONTROL DE LA ENTRADA/SALIDA

Cada dispositivo periférico del sistema está controlado mediante uno o más ficheros. Estos ficheros se tratan de la misma forma que los ficheros de datos ordinarios. En otras palabras, para mostrar un carácter en la pantalla, el carácter se escribe en el fichero asociado a ésta. Esto simplifica la entrada/salida y oculta el usuario las peculiaridades de los distintos dispositivos.

GESTIÓN DE LOS DISPOSITIVOS DE ALMACENAMIENTO MASIVO

Todo el volumen de almacenamiento masivo disponible se divide en ficheros. El contenido de un fichero puede estructurarse de acuerdo a los deseos del usuario, pero las relaciones entre ficheros están controladas a nivel de sistema mediante directorios. Los directorios permiten agrupar los ficheros y estructurar los métodos para dar acceso a ciertos ficheros a los distintos usuarios. El SO mantiene un directorio raíz, a través del cual se llega a todos los demás ficheros del sistema. El sistema de ficheros se considera una de las características más importantes de UNIX.

PROTECCIÓN

La protección dentro del sistema se logra mediante bits asociados a cada uno de los ficheros. Estos permiten otorgar permisos de lectura, escritura y ejecución, tanto para el propietario del fichero como para los demás usuarios. Diferentes usuarios pueden tener diferentes niveles de privilegios, que determinan la extensión con la que pueden acceder a los ficheros y por tanto utilizar los recursos del ordenador.

El sistema dispone de mecanismos de tratamiento de errores, de modo que si se detecta un error durante la ejecución de un programa el SO salta a una subrutina de tratamiento de errores. Además el usuario dispone de la facilidad de interrumpir un programa cuya ejecución no parezca correcta.

INTERFAZ CON EL USUARIO

Todos las órdenes del SO son interpretados por un programa denominado shell (concha, para dar la idea que es el caparazón o la parte visible del sistema). Una característica notable del *shell* es que permite al usuario controlar la extensión del funcionamiento en multitarea del ordenador. En circunstancias normales el SO completa la ejecución de una orden antes de indicar al usuario que puede introducir otro. Sin embargo, un usuario puede pedir al SO que comience a trabajar con una orden a la vez que queda dispuesto para aceptar otro. Las acciones derivadas de ambas órdenes se ejecutan juntas en multiprogramación.
En UNIX también se han producido importantes avances en materia de GUI (X-Windows) y además, con la integración de la red en el SO se han desarrollado

múltiples aplicaciones relacionadas con la distribución de información (WWW = World Wide Web) que sean accesibles desde aplicaciones que se ejecutan en otros ordenadores con otros SO (p.e. desde PC bajo MSDOS)
En la actualidad hay UNIX para PC cuyo código y manuales pueden obtenerse por red gratuitamente al ser software de dominio público. Quizá el más popular es el LINUX.

2.4.7.3 SISTEMAS OPERATIVOS PARA MAINFRAMES

Las grandes máquinas, han venido históricamente usando sistemas operativos especiales para cada una de ellas, sin embargo ésto está cambiando afortunadamente. Así, en la actualidad la mayoría incorporan el concepto de máquina virtual y podemos describir algún SO suficientemente representativo de los que utilizan este tipo de ordenador. Un ejemplo característico es el VME/B, que se diseñó y desarrolló a la vez que los ordenadores para los que está pensado (La serie 2900 de ICL).

CARACTERÍSTICAS GENERALES

El VME/B es un SO de propósito general muy sofisticado que incorpora varias características modernas. Soporta multiacceso, *batch* y tratamiento de transacciones. Uno de sus primeros objetivos es el de proporcionar un interfaz de alto nivel simple y coherente para todos los usuarios. Es flexible, en el sentido de que puede ser adaptado a los requisitos de una instalación particular. El VME/B está escrito en un lenguaje de alto nivel especialmente diseñado para ese propósito. Las comunicaciones con el VME/B se realizan mediante un lenguaje de control del sistema (SCL) que está estructurado como un lenguaje de alto nivel.

La máquina virtual disponible para cada usuario incluye sus programas y sus datos, así como todas las facilidades del SO y las utilidades que necesite el usuario. Para un usuario la máquina virtual aparece como una entidad totalmente autocontenida, aunque el VME/B permite compartir determinadas áreas de datos y código para evitar la duplicación del *software*. Se utiliza un sistema de protección muy sofisticado para garantizar la seguridad de cada una de las máquinas virtuales y de los segmentos compartidos.

ESTRUCTURA DEL SISTEMA

Como muchos otros SO, el VME/B está constituido por una serie de capas. En el nivel más interno se sitúa el núcleo, que es el encargado de transformar el *hardware* del ordenador en un conjunto de máquinas virtuales. La siguiente capa está constituida por el *software* del sistema, que gestiona los recursos de las máquinas virtuales. La capa más externa contiene los programas de aplicación a

diferencia de otros SO más antiguos, el VME/B no presenta una separación rígida entre el *software* del sistema y los programas de aplicación.

SISTEMA DE GESTIÓN DE MEMORIA Y COMUNICACIONES

Cada una de las máquinas virtuales que corren bajo VME/B dispone de un área de almacenamiento virtual mucho mayor que la memoria principal del ordenador. El VME/B controla todas las transferencias de segmentos de programas y datos desde y hacia los medios de almacenamiento masivo para que el sistema de memoria virtual sea una realidad.

Uno de sus objetivos generales es el proporcionar un eficiente servicio de comunicaciones. El aspecto *hardware* del problema se resuelve utilizando una serie de controladores de entrada/salida que enlazan los dispositivos periféricos con la memoria principal. El aspecto *software* está controlado por el VME/B , que controla y planifica todas las entradas/salidas. Así se libera a los usuarios de los problemas que plantean las comunicaciones y se hace un mejor uso de los recursos del ordenador.

GESTIÓN DEL ALMACENAMIENTO MASIVO

Todos los aspectos relativos al uso y la seguridad de los ficheros corren a cargo de un sistema de almacenamiento integrado que forma parte del VME/B. El usuario puede tomar la decisión de hacerse responsable del control del estado de sus ficheros en el sistema de almacenamiento masivo o dejar esta tarea bajo el control del SO.

PROTECCIÓN

Uno de los aspectos principales del VME/B es su sistema de protección que está basado en el concepto de niveles de privilegio. Cada proceso que se ejecuta en el sistema tiene asociado un nivel de privilegio, de 0 a 15, siendo el nivel 0 el de mayor privilegio y el nivel 15 el más bajo. Los niveles 0 a 2 están reservados para el núcleo; los niveles 3 a 9, para el *software* del sistema, y los niveles 10 a 15, para los programas de aplicación.

Cuando un proceso se está ejecutando, un registro, denominado registro de control de acceso, contiene su nivel de privilegio. Cada uno de los segmentos de datos a los que el proceso pueda acceder tiene una clave de acceso para lectura y una clave de acceso para escritura. Cada clave tiene un valor entre 0 y 15. Sólo se permite el acceso si el nivel de privilegio del registro de control de acceso es menor o igual al valor de la clave. Por ejemplo, un proceso con nivel de privilegio 11 puede leer de un segmento con clave de acceso 14, pero no puede escribir en un segmento con clave de acceso 8. Además cada segmento dispone de un bit de autorización de

ejecución. Sólo si éste tiene el valor 1 se puede ejecutar el código contenido en ese segmento.

2.5. SOPORTE LÓGICO DE LAS REDES DE COMPUTADORES

Una vez enfatizados los módulos y programas del soporte lógico de un computador, que facilitan el proceso de programación, en este epígrafe trataremos la parte del soporte lógico que concierne al uso de las redes de computadores, ya que éstas, en la actualidad, han revolucionado el uso de la tecnología computacional. Aplicaciones que dependían de un sistema centralizado con una macrocomputadora y una serie de varias terminales, en la actualidad usan redes, en las cuales cada usuario cuenta con un ordenador propio.

2.5.1 SOPORTE LÓGICO BÁSICO

2.5.1.1 PAQUETES

De la misma forma que cuando alguien escribe una carta, envía el texto metiéndolo en un sobre, generalmente de tamaño estándar. Un principio similar se usa en muchos sistemas de comunicación. Los datos se transmiten, no carácter a carácter, ni en grupos de caracteres, sino en **paquetes**. Un paquete es un conjunto de datos para transmitir, que se incluyen en una entidad mayor, en la que se encuentran datos para el correcto enrutamiento del paquete por la red de comunicaciones sobre la que se mueve, así como mecanismos (bastante sofisticados por cierto) para la detección y corrección de errores. Los datos que se incluyen en los paquetes siguen unas normas estrictas, ya que todos los dispositivos de una red envían y reciben los datos, utilizando paquetes del mismo tipo. Los paquetes se han transformado en la unidad de transmisión y recepción de datos dentro de muchas redes de comunicación de datos (denominadas **redes de conmutación de paquetes**).

2.5.1.2 PROTOCOLOS DE RED

El tipo y topología de una red establecen su estructura básica, pero aún así, cada computadora necesita el hardware para transmitir y recibir información. El dispositivo que lleva a cabo esta función es la **tarjeta de interfaz de red (NIC = network interface card**). La NIC es un tipo de tarjeta de circuitos impresos, que se instala en una de las ranuras de expansión de la computadora y proporciona un puerto en la parte trasera de la PC al cual se conecta el cable de la red. La computadora también requiere del software que le indique cómo usar la NIC.

Ambos, el software de la red y la NIC, se tienen que adherir al protocolo de la red, que es el conjunto de estándares para la comunicación. Un **protocolo de red** es como un lenguaje para la comunicación de información ya que para que dos máquinas intercambien información, deben hablar el mismo idioma. Como ocurre a menudo, los protocolos están en continuo estado de cambio, ya que cada vez que aparece un nuevo estándar, alguien inventa otro que hace el trabajo más rápido y en forma más fiable. Veamos los protocolos mas comunes (cada uno para cierta topología de red y con ciertas características estándar):

2.5.1.2.1 Ethernet

Ethernet es actualmente el protocolo de red más común, que al basarse en la topología de bus lineal resulta de bajo costo y relativamente simple. Sin embargo, con un bus lineal, cada nodo de la red debe tomar su turno para enviar información y al igual que en una línea telefónica, que solo una persona puede usarla a la vez, cada vez que una máquina necesite enviar información a otra, primero deberá revisar si la red está disponible. Si la red está siendo utilizada por otra estación, espera brevemente y lo intenta de nuevo; como se adivina, cuando haya muchas computadoras en una red Ethernet, el tiempo de acceso se hace notoriamente lento.

Afortunadamente existen configuraciones que permiten utilizar el Ethernet en bus lineal *lógico* en lugar de uno lineal *físico*, llamado **10 base-T**, que utiliza un equipo que proporciona las ventajas de una topología en estrella centralizada con la flexibilidad y capacidad de un bus lineal. En cualquier caso los buses de la red Ethernet están limitados a 2 500 pies (762 metros) de distancia y corren a 10 Mbits por segundo.

2.5.1.2.2 Token Ring

El protocolo de red **Token Ring**, como indica su nombre, se basa en la topología de anillo. Su característica consiste en que tiene un hardware de control que transmite direcciones electrónicas, por la red, muchas veces por segundo. Cada nodo examina estas direcciones para determinar si alguna le pertenece. En caso negativo, las direcciones siguen viajando por la red sin alteración. Si la dirección concuerda, el nodo puede anexarle un paquete de información y hacerlo llegar a otro nodo de la red y, a la inversa un ordenador de la red puede leer el contenido de un paquete relacionado con su dirección. Este proceso de hacer circular un grupo de paquetes dirigidos a cada estación para cogerlos o pasarlos, simula la acción de pasar una señal (token), de ahí el nombre de Token Ring.

Las redes **Token-Ting**, aunque caras, tienen la ventaja de que la información viaja de una manera controlada a través del anillo en una dirección. Con este enfoque, la información no puede chocar y la red puede operar a mayores velocidades (probablemente pronto veamos velocidades de hasta 100 Mbits por segundo).

2.5.1.2.3 ARCNET

La **ARCNET** se basa en la topología de estrella o estrella distribuida, con una topología y protocolo propios, utiliza cable coaxial y la estrella es perpetuada mediante el uso de paneles de control conectados a la red. La ARCNET es mas lenta, cerca de 2.5 Mbits por segundo, aunque barata, confiable y fácil de instalar y expandir.

2.5.1.3 MODELO DE CAPAS ISO

Una de los metas principales para la interconexión de distintos equipos informáticos (diversos fabricantes, modelos, etc.) es conseguir la utilización de sistemas y procedimientos de intercambio de información comunes. En definitiva, el primer problema es establecer los interfaces y protocolos de interconexión, para ser fijados existen asociaciones internacionales especializadas como la ISO (International Standard Organization). Esta ha establecido una arquitectura para los sistemas de comunicación (Modelo ISO), que utiliza la sistemática de considerar el problema de análisis y diseño de redes de computadoras por **capas** (o **niveles**). Así para la interconexión de dos equipos informáticos, A y B, se definen 7 niveles, que se encargan de llevar a cabo la transmisión (Ver Figura 2.6). Cada nivel "proporciona" o "se relaciona" con el nivel superior por medio de interfaces, y las relaciones entre los dos equipos en cada nivel se efectúan a través de protocolos. En esencia, en cada capa o nivel i:

a) se diseña el protocolo nivel i (conexión Ai, Bi), utilizando las interfaces proporcionadas por el nivel inferior (i-1).(Niveles Ai - 1 y Bi - 1);

b) se diseña la interfaz de nivel i como servicio para el nivel i+1 (relaciones Ai con Ai + 1, y Bi con Bi + 1), utilizando las interfaces proporcionadas por el nivel inferior (i-1) (Niveles Ai - 1 y Bi - 1).

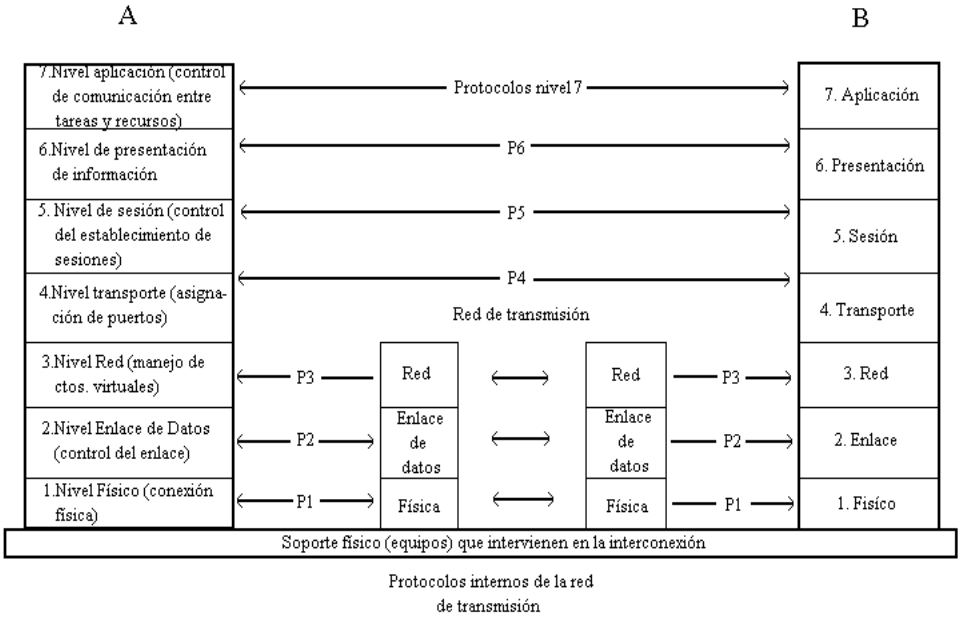

Fig.2.6. Esquema de protocolos en el Modelo de capas ISO para la interconexión de dos sistemas A y B.

En la Figura 2.6, puede verse un esquema simplificado del Modelo de referencia ISO En los extremos figuran los sistemas entre los que se requiere efectuar la conexión (Sistemas A y B). En el centro figuran otros elementos que intervienen sólo con objeto de efectuar la transmisión (fundamentalmente equipos terminales de las líneas de comunicaciones e interfaces de procesadores de comunicaciones). Las capas o niveles conceptuales del modelo ISO son las siguientes:

Nivel 1: CAPA FÍSICA. En este nivel se especifican los parámetros mecánicos, eléctricos, etc., de la interconexión entre el equipo terminal de datos y el equipo terminal de la línea de comunicaciones. Las unidades de información son bits y su misión es asegurar que si un emisor envía un 1 al receptor le llega un 1. Evidentemente, su relación con el soporte lógico del ordenador, no es muy relevante.

Nivel 2. CAPA DE ENLACE DE DATOS. Es la capa, encargada de transmitir sin errores, ya no bits, sino bloques de información (llamadas tramas) entre dos puntos físicos de la red. Para realizar esto, distribuye las cadenas de bits proporcionadas por el nivel 1 en tramas de datos, emitiéndolas secuencialmente y comprobando si hay errores en la transmisión.

Nivel 3. CAPA DE RED. Este nivel controla las operaciones en la red de transmisión. Los elementos de información que trata son paquetes, que están

compuestos de tramas. Su responsabilidad por tanto es la de planificar el trayecto que deben seguir los paquetes en el interior de la red y de controlar la conmutación de los circuitos.

Nivel 4. CAPA DE TRANSPORTE. Se encarga del transporte eficiente de la información, desde la fuente el destino, a través de la red de conmutación de paquetes. En este nivel, al igual que los de mayor nivel, las interacciones (protocolos) se realizan entre los equipos informáticos fuente y destino, frente a los protocolos de los niveles 1, 2 y 3 que se realizan entre equipos de nodos vecinos, sean o no fuente o destino. Las unidades de información que considera son mensajes , que están constituidos por paquetes.

Nivel 5. CAPA DE SESION. Cada vez que se desea efectuar una comunicación entre dos sistemas o usuarios, debe establecerse una sesión de comunicaciones entre ambos. Este nivel es el responsable de la realización y control del diálogo entre procesos de distintos nudos: establecer la comunicación, sincronizar los diálogos, cerrar la sesión, etc. Para establecer una sesión el usuario debe proporcionar una dirección del equipo remoto al cual pretende conectarse (dirección de sesión). Este nivel, se encarga de transformar las direcciones de sesiones en direcciones de transporte, facilitándolas al nivel 4.

Nivel 6. CAPA DE PRESENTACIÓN. Es el nivel responsable de la forma (o formato) de la estructura de datos intercambiados entre los procesos que dialogan. Se encarga de: interpretar las estructuras de las informaciones intercambiadas por los procesos de la aplicación, gestionar las terminales, transferir los archivos, comprimir los datos y en su caso de las transformaciones criptográficas que hubiera. Esta capa trata de homogeneizar los formatos de intercambio de información entre equipos de la red al objeto de facilitar las tareas de la capa de sesión.

Nivel 7. CAPA DE APLICACIÓN. Este nivel es el que está en contacto con el usuario de toda la red, únicamente “ve” o “se las entiende” con esta capa. Los usuarios utilizan aplicaciones informáticas , las cuales al ejecutarse , quedan constituidas por conjuntos de tareas que requieren determinados recursos. Tanto estas tareas como los recursos , se hallan distribuidas a lo largo y ancho de la red. El nivel de aplicación se encarga de las funciones específicas de intercambio y cooperación tanto entre los procesos como entre los recursos , que utilizan la aplicación.

2.5.2 RELACIONES EN UNA RED

Al describir una red como LAN (Local Area Network) o WAN (Wide Area Network) se define el área geográfica que la red cubre. El segundo escalón es

describir cómo las computadoras individuales ó nodos, pueden interactuar con otras computadoras en la red , esto es de que manera están organizadas.

2.5.2.1 LA RELACIÓN CLIENTE-SERVIDOR

Una manera de organizar redes es llamada **cliente-servidor**, una estrategia jerárquica en la cual una computadora en particular sirve a las necesidades de almacenamiento, y algunas veces a las necesidades de proceso, de todos los nodos de la red. El tipo más común de instalación de cliente-servidor es una LAN, compuesta de PC o estaciones de trabajo conectadas a un **servidor de red**, el cual puede o no, también ser usado como el dispositivo de almacenamiento principal de la red. Un programa **cliente** corriendo en uno de los nodos puede requerir información específica del servidor. El programa del servidor puede traer la información solicitada de sus bases de datos y pasársela al cliente.

2.5.2.2 COMPUTACIÓN PAR A PAR

Otra distribución es la computación par a par, una estrategia de red en la cual las computadoras pueden actuar como cliente y servidores a la vez. En otras palabras, cada nodo tiene acceso a todos o a algunos de los recursos de los otros nodos. Por ejemplo, con la versión de Windows para trabajo en grupo, los usuarios tienen acceso a los discos duros e impresoras conectados a otras computadoras del grupo de trabajo.

Una LAN par a par permite a los usuarios compartir periféricos, incluyendo el disco de almacenamiento, de tal forma que pueden tener acceso a la misma información y a los mismos programas. Algunas redes par a par, de alto rendimiento, tales como las redes de computadoras Unix, permiten la computación distribuida; esto es, permite que el usuario de una máquina tenga a su disposición la capacidad de proceso de otras computadoras de la red. Eso significa que se puede transferir aquellas tareas que requieran muchos recursos de CPU, a computadoras que en este momento estén disponibles, liberando a su propia máquina.

2.5.3 PROGRAMAS DE APLICACIONES DE LAS COMUNICACIONES

Desde hace años los usuarios han reconocido el valor de intercambiar datos y software. La respuesta a esta demanda vino dada por Hayes Microcomputer Products, quien desarrolló en 1978, el Smartmodem, el primer módem para computadoras personales. Con él se introdujo una nueva industria que actualmente

proporciona toda clase de servicios para ordenadores con módem, cosa que ha hecho que su soporte lógico se tuviera que adaptar a estas nuevas posibilidades.

Correo electrónico.-

Una de las aplicaciones de mayor alcance en las comunicaciones de datos es el **correo electrónico (e-mail),** un sistema para intercambiar mensajes escritos (y crecientemente, mensajes de voz) a través de una red. E-mail es algo así como una mezcla entre el sistema postal y un contestador telefónico, donde cada usuario tiene una dirección única. Para enviar un mensaje e-mail a alguien, se teclea la dirección del destinatario y después el mensaje. Cuando se termina, el mensaje se envía a la dirección. Cuando aquél acceda al sistema e-mail, se le informa que le ha llegado correo. Después de leer el mensaje, el destinatario puede guardarlo, borrarlo, hacerlo llegar a alguien más o responderlo enviando un mensaje de respuesta. Además de enviar texto, muchos sistemas permiten anexar información como hojas de cálculo o documentos como complemento al mensaje. Una red local puede tener conexión a las grandes redes de información, lo que proporciona al usuario de red acceso por correo a literalmente millones de usuarios e-mail en todo el mundo.

El correo electrónico es eficiente y de bajo costo. Los usuarios pueden enviar mensajes escritos sin preocuparse de que el destinatario esté o no usando la computadora en ese momento. A través de las redes centralizadas, el mensaje es remitido casi instantáneamente y además, una vez que se ha instalado la red, el correo electrónico es muy barato.

Computadoras de acceso remoto.-

Uno de los grandes beneficios de un módem es que permite acceder a computadoras remotas desde cualquier teléfono, no importa donde se encuentre. Al marcar con un módem el número de la red se puede obtener información, acceder a archivos e intercambiar mensajes por correo electrónico. Con una ordenador y un módem, todo esto es posible y el ordenador debe estar preparado para esta nueva situación, a fin de obtener las ventajas y evitar los inconvenientes que pueden darse.

Transferencia de archivos.-

Por transferencia de archivos entenderemos, simplemente, enviar un archivo de una computadora a otra y es la aplicación mas frecuente de los módem. Para que un archivo pueda ser transferido de una computadora a otra, ambas deben utilizar el mismo protocolo de transferencia de archivos. Como sus homólogos de redes, los protocolos de transferencia de archivos deben ser acordados previamente (por ejemplo Kermit, Xmodem etc.). Una de sus funciones de más importancia es

revisar los errores mientras se está enviando el archivo. Normalmente, la comunicación del módem es bidireccional, lo cual significa que la computadora que recibe puede responder a la que envía para asegurarse de que la información que recibe no contiene errores. Si existe algún error, la computadora que envía la información, transmite de nuevo, cualquier parte que sea incorrecta.

Internet.-

Internet es una enorme red de redes que se enlaza a muchas de las redes científicas, y de investigación alrededor del mundo, así como a un número creciente de redes comerciales. La Internet, a menudo llamada " la red" ("the net"), fue iniciada en 1969 por el Departamento de Defensa de EEUU y creció gradualmente hasta convertirse, en una red mundial para investigación científica. Ahora es mucho más que eso. La mayoría de la universidades están conectadas a la Internet, así como muchas compañías y la mayoría de compañías que proporcionan servicios de información en línea.

Es difícil definir a la Internet, de hecho es como un gran servicio de información, que ofrece correo electrónico, boletines electrónicos y servicios de acceso de información que ponen a disposición del usuario directorios de archivos y bases de datos a nivel mundial. Se puede decir que la Internet es la más poderosa herramienta de investigación y búsqueda jamás creada y está creciendo a pasos agigantados. Por ello no hay que extrañarse, que los nuevos soporte lógicos para ordenadores, como Windows 95, incluyan ya el acceso a Internet como una de las utilidades incluidas en sus características.