P R O L O G O

El presente trabajo trata sobre la Arquitectura de Windows NT. La investigación se ha llevado a cabo desde cuatro puntos de vista que son los componentes señalados por Andrew Tanenbaum para el enfoque del estudio de un Sistema Operativo, es decir: procesos, administración de la memoria, ficheros y entrada/salida. Se ha comenzado con un capítulo dedicado a conceptos básicos y otro a la arquitectura global del sistema, para seguidamente estudiar en detalle cada uno de los cuatro puntos antes citados.

Se puede decir que Windows NT representa la primera propuesta de la empresa Microsoft por construir un Sistema Operativo serio, capaz de competir con los dinosaurios clásicos como UNIX o VMS (o al menos intentarlo). Para ello se ha elegido un enfoque cliente−servidor, con la intención de alcanzar el sueño de los Sistemas Distribuidos.

Windows NT no se puede considerar todavía un sistema operativo de carácter doméstico. Pero el objetivo de sus diseñadores parece ser que lo sea dentro de muy pocos años, y de hecho, para que los programadores vayan entreteniéndose en el uso de las llamadas al sistema (el llamado Win32), han construido un Sistema Operativo de poco confiable. Hablamos, cómo no, de Windows 95.

La arquitectura de Windows 95/98 no tiene absolutamente nada que ver con la de Windows NT. No es que sea un mal trabajo pero deja bastante que desear.

Durante la elaboración de este trabajo nos hemos encontrado con varias dificultades principalmente por la falta de fuentes bibliográficas que trataran el diseño con la suficiente profundidad como para satisfacer el nivel que se buscaba. Los libros que se encontraron, la mayoría se refieren a la Administración del Sistema a alto nivel, o bien de la programación bajo Windows NT en lenguaje C++, por lo que he tenido que recurrir a diversos artículos de la Internet, principalmente al nodo de Microsoft en Internet. Por ello, pedimos disculpas por las posibles erratas que puedan existir.

Para tratar ciertos temas dentro de cada capítulo se ha creído conveniente apoyarnos en la descripción de las llamadas al sistema, ya que, a nuestro entender, de esta manera todo resulta mucho más fácil de explicar y de comprender.

T A B L A D E C O N T E N I D O S

L I S T A D E G R A F I C O S

A R Q U I T E C T U R A D E L S I S T E M A O P E R A T I V O W I N D O N S N T

I. INTRODUCCION Y CONCEPTOS BASICOS

- **Eventos A Través Del Tiempo**

A finales de los años 40's y a principios de los años 50's las computadoras masivas, eran controladas por tubos al vacío inestables. Todo la programación se hacía directamente en lenguaje de máquina porque la industria no había avanzado lo suficiente para necesitar Sistemas Operativos. Con la aparición del transistor a mediados de los 50's, las computadoras se fueron haciendo más y más confiables.

Lenguajes crudos como Ensamblador y Fortran aparecieron, pero un Sistema Operativo (S.O.), tal como los conocemos ahora, aún no. Para accesar a la programación de la maquinaria se manejaron tarjetas perforadas.

1960's. Cuando IBM introdujo la computadora System/360 intentó tomar el mercado científico y el comercial. Cuando en este proyecto surgieron problemas de conflictos por la arquitectura, se inició el desarrollo de un software que resolviera todos aquellos conflictos, el resultado fue un muy complejo sistema operativo. Luego AT&T trató de desarrollar a Multics, un Sistema Operativo que soportara cientos de usuarios de tiempo compartido, pero falló. Más adelante científicos de la computación desarrollaron Unics, que sería monousuario. Ello marca el nacimiento de Unix (1969), el primero de los sistemas operativos modernos.

1980's. En este tiempo la arquitectura de las computadoras, circuitos LSI (Large Scale Integration) abrieron el paso para una nueva generación de computadoras. DOS de Microsoft aparece en 1981 dominando este mercado de las PCs inmediatamente, aunque el sistema UNIX, predomina en las estaciones de trabajo.

1990's. Aumenta el uso de conexiones en redes, equipos de trabajo y aplicaciones distribuidas, los cuales surgen en la década anterior, con ello los Sistemas Operativos como Unix, Windows NT, etc., soportan muchos clientes, dando así el nacimiento de la Computación en Red.

- **Sistema Operativo**
- **Introducción**

Software básico que controla y administra los recursos de una computadora. El sistema operativo tiene tres grandes funciones: coordina y manipula el hardware de la computadora, como la memoria, las impresoras, las unidades de disco, el teclado o el mouse; organiza los archivos en diversos dispositivos de almacenamiento, como discos flexibles, discos duros, discos compactos o cintas magnéticas, y gestiona los errores de hardware y la pérdida de datos.

- **¿Cómo funciona un sistema operativo?**

Los Sistemas Operativos controlan diferentes procesos de la computadora. Un proceso importante es la interpretación de los comandos que permiten al usuario comunicarse con el ordenador. Algunos intérpretes de

instrucciones están basados en texto y exigen que las instrucciones sean tecleadas. Otros están basados en gráficos, y permiten al usuario comunicarse señalando y haciendo clic en un icono.

Los Sistemas Operativos pueden ser de tarea única o multitarea. Los sistemas operativos de tarea única, más primitivos, sólo pueden manejar un proceso en cada momento. Por ejemplo, cuando la computadora está imprimiendo un documento, no puede iniciar otro proceso ni responder a nuevas instrucciones hasta que se termine la impresión.

Todos los Sistemas Operativos modernos son multitarea y pueden ejecutar varios procesos simultáneamente. En la mayoría de los ordenadores sólo hay una UCP; un Sistema Operativo multitarea crea la ilusión de que varios procesos se ejecutan simultáneamente en la UCP. El mecanismo que se emplea más a menudo para lograr esta ilusión es la multitarea por segmentación de tiempos, en la que cada proceso se ejecuta individualmente durante un periodo de tiempo determinado. Si el proceso no finaliza en el tiempo asignado, se suspende y se ejecuta otro proceso. Este intercambio de procesos se denomina conmutación de contexto. El sistema operativo se encarga de controlar el estado de los procesos suspendidos. También cuenta con un mecanismo llamado planificador que determina el siguiente proceso que debe ejecutarse. El planificador ejecuta los procesos basándose en su prioridad para minimizar el retraso percibido por el usuario. Los procesos parecen efectuarse simultáneamente por la alta velocidad del cambio de contexto.

Los Sistemas Operativos pueden emplear memoria virtual para ejecutar procesos que exigen más memoria principal de la realmente disponible. Con esta técnica se emplea espacio en el disco duro para simular la memoria adicional necesaria.

- **Sistemas Operativos actuales**

Los sistemas operativos empleados normalmente son UNIX, Macintosh OS, MS−DOS, OS/2 y Windows−NT. El UNIX y sus clones permiten múltiples tareas y múltiples usuarios. Su sistema de archivos proporciona un método sencillo de organizar archivos y permite la protección de archivos. Sin embargo, las instrucciones del UNIX no son intuitivas. Otros sistemas operativos multiusuario y multitarea son OS/2, desarrollado inicialmente por Microsoft Corporation e International Business Machines (IBM) y Windows−NT, desarrollado por Microsoft. El sistema operativo multitarea de las computadoras Apple se denomina Macintosh OS. El DOS y su sucesor, el MS−DOS, son sistemas operativos populares entre los usuarios de computadoras personales. Sólo permiten un usuario y una tarea.

- **Sistema Operativo de Red**

A un Sistema Operativo de Red se le conoce como NOS. Es el software necesario para integrar los muchos componentes de una red en un sistema particular, al cual el usuario final puede tener acceso.

Otra definición es la siguiente; es un software que rige y administra los recursos, archivos, periféricos, usuarios, etc., en una red y lleva el control de seguridad de los mismos.

Un NOS maneja los servicios necesarios para asegurar que el usuario final tenga o esté libre de error al accesar a la red. Un NOS normalmente provee una interfaz de usuario que es para reducir la complejidad y conflictos al momento de usar la red.

Dentro del contexto del Sistema Operativo de Red, se pueden escribir aplicaciones tales como un sistema de correo electrónico pueden ser escritas para que permitan "conexiones virtuales" entre entidades de red, sin intervención humana directa.

- **Diferencia entre un S.O. Distribuido, un S.O. de Red y un S.O. Centralizado.**

En un Sistema Operativo de Red, los usuarios saben de la existencia de varias computadoras y pueden conectarse con máquinas remotas y copiar archivos de una máquina a otra, cada máquina ejecuta su propio sistema operativo local y tiene su propio usuario o grupo de usuarios.

Por el contrario, un Sistema Operativo Distribuido es aquel que aparece ante sus usuarios como un sistema tradicional de un solo procesador, aun cuando esté compuesto por varios procesadores. En un sistema distribuido verd adero, los usuarios no deben saber del lugar donde su programa se ejecute o del lugar donde se encuentran sus archivos; eso debe ser manejado en forma automática y eficaz por el Sistema Operativo.

Además son sistemas autónomos capaces de comunicarse y cooperar entre sí para resolver tareas globales. Es indispensable el uso de redes para intercambiar datos. Además de los servicios típicos de un Sistema Operativo, un Sistema Distribuido debe gestionar la distribución de tareas entre los diferentes nodos conectados. También, debe proporcionar los mecanismos necesarios para compartir globalmente los recursos del sistema.

Sistemas Operativos Centralizados, de un solo procesador, de un solo CPU o incluso tradicionales; en todo caso, lo que esto quiere decir es que un sistema operativo controla una sola computadora.

II. DISEÑO DE WINDOWS NT

Windows NT presenta una arquitectura del tipo cliente−servidor. Los programas de aplicación son contemplados por el sistema operativo como si fueran clientes a los que hay que servir, y para lo cual viene equipado con distintas entidades servidoras.

Uno de los objetivos fundamentales de diseño fue el tener un núcleo tan pequeño como fuera posible, en el que estuvieran integrados módulos que dieran respuesta a aquellas llamadas al sistema que necesariamente se tuvieran que ejecutar en modo privilegiado (también llamado modo kernel, modo núcleo y modo supervisor). El resto de las llamadas se expulsarían del núcleo hacia otras entidades que se ejecutarían en modo no privilegiado (modo usuario), y de esta manera el núcleo resultaría una base compacta, robusta y estable. Por eso se dice que Windows NT es un sistema operativo basado en micro−kernel.

Es por ello que en un primer acercamiento a la arquitectura distinguimos un núcleo que se ejecuta en modo privilegiado, y se denomina Executive, y unos módulos que se ejecutan en modo no privilegiado, llamados subsistemas protegidos.

Los programas de usuario (también llamados programas de aplicación) interaccionan con cualquier sistema operativo (SO) a través de un juego de llamadas al sistema, que es particular de cada SO. En el mundo Windows en general, las llamadas al sistema se denominan API (Application Programming Interfaces, interfaces para la programación de aplicaciones). En Windows NT y en Windows 95 se usa una versión del API llamada API Win32. Un programa escrito para Windows NT o Windows 95, y que por consiguiente hace uso del API Win32, se denomina genéricamente "programa Win32", y de hecho esta denominación es bastante frecuente en artículos y libros al respecto. Desgraciadamente, y conviene dejarlo claro cuanto antes, el término "Win32" tiene tres acepciones (al menos hasta ahora) totalmente distintas. Una es el API, otra es el nombre de uno de los subsistemas protegidos de Windows NT del que hablaremos más adelante, y por último se denomina Win32s a una plataforma desarrollada por Microsoft, similar a Windows 3.1, pero que usa el API Win32 en vez del API Win16 del Windows 3.1.

Hechas estas aclaraciones, podemos continuar adelante. Algunas de las llamadas al sistema, debido a su naturaleza, son atendidas directamente por el Executive, mientras que otras son desviadas hacia algún subsistema. Esto lo veremos con detalle en breve.

El hecho de disponer de un núcleo rodeado de subsistemas que se ejecutan en modo usuario nos permite

además añadir nuevos subsistemas sin producir ningún tipo de confrontación.

En el diseño de Windows NT han confluido aportaciones de tres modelos: el modelo cliente−servidor, el modelo de objetos, y el modelo de multiprocesamiento simétrico.

- **Modelo cliente−servidor**. En la teoría de este modelo se establece un kernel que básicamente se encarga de recibir peticiones de procesos clientes y pasárselas a otros procesos servidores, ambos clientes y servidores ejecutándose en modo usuario. Windows NT pone el modelo en práctica pero no contempla el núcleo como un mero transportador de mensajes, sino que introduce en él aquellos servicios que sólo pueden ser ejecutados en modo kernel. El resto de servicios los asciende hacia subsistemas servidores que se ejecutan en modo usuario, independientes entre sí, y que por tanto pueden repartirse entre máquinas distintas, dando así soporte a un sistema distribuido (de hecho, el soportar los sistemas distribuidos fue otra de las grandes directivas de diseño de este SO).

- **Modelo de objetos**. Decir que no implementa puramente la teoría de este modelo, sino que más bien lo que hace es simplemente contemplar los recursos (tanto internos como externos) como objetos. Más adelante daremos una lista de los objetos de Windows NT. Brevemente, señalar que todo objeto ha de poseer identidad propia (es único y distinguible de todos los demás), y una serie de atributos (variables) y métodos (funciones) que modifican sus atributos. Los objetos interaccionan entre sí a través del envío de mensajes. No sólo existen en Windows NT objetos software (lógicos), sino que los dispositivos hardware (físicos) también son tratados como objetos (a diferencia de UNIX, que recordemos trataba a los dispositivos como ficheros).

- **Modelo de multiprocesamiento simétrico**. Un SO multiproceso (o sea, aquel que cuenta con varias CPU y cada una puede estar ejecutando un proceso) puede ser simétrico (SMP) o asimétrico (ASMP). En los sistemas operativos SMP (entre los que se encuentran Windows NT y muchas versiones de UNIX) cualquier CPU puede ejecutar cualquier proceso, ya sea del SO o no, mientras que en los ASMP se elige una CPU para uso exclusivo del SO y el resto de CPU quedan para ejecutar programas de usuario. Los sistemas SMP son más complejos que los ASMP, contemplan un mejor balance de la carga y son más tolerantes a fallos (de manera que si un subproceso del SO falla, el SO no se caerá pues podrá ejecutarse sobre otra CPU, cosa que en los ASMP no sería posible, con lo que se bloquearía el sistema entero).

Comencemos describiendo los subsistemas protegidos, para seguidamente estudiar la estructura del Executive.

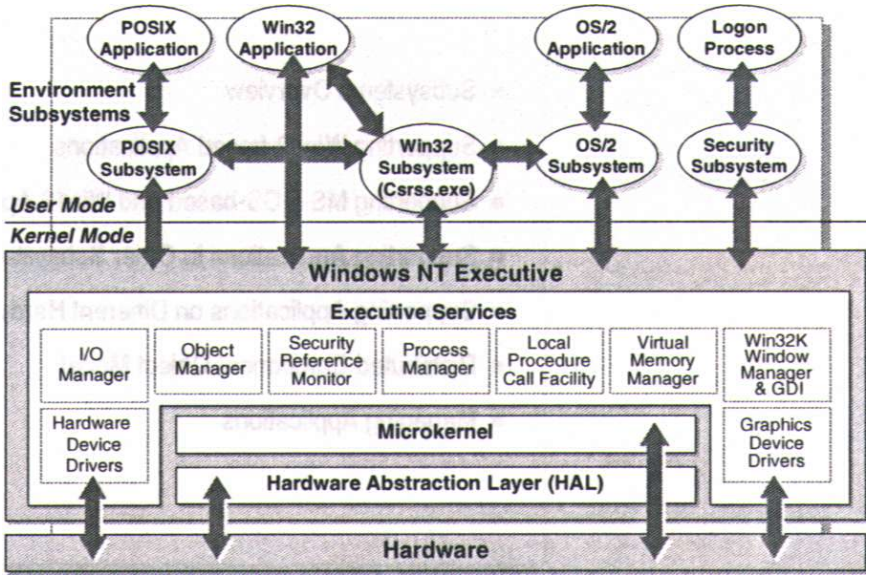

Figura 1. El núcleo se ejecuta en modo privilegiado (Executive) y en modo no privilegiado (subsistemas protegidos)

- **Los Subsistemas Protegidos**

Son una serie de procesos servidores que se ejecutan en modo usuario como cualquier proceso de usuario, pero que tienen algunas características propias que los hacen distintos. Al decir subsistemas protegidos nos referiremos, pues, a estos procesos. Se inician al arrancar el SO. Los hay de dos tipos: integrales y de entorno.

- **Un Subsistema Integral:** es aquel servidor que ejecuta una función crítica del SO (como por ejemplo el que gestiona la seguridad). Tenemos los siguientes:

    - **El Subsistema Proceso de Inicio (Logon Process)**

El proceso de inicio (Logon Process) recibe las peticiones de conexión por parte de los usuarios. En realidad son dos procesos, cada uno encargándose de un tipo distinto de conexión:

- El proceso de inicio local: gestiona la conexión de usuarios locales directamente a una máquina Windows NT.
- El proceso de inicio remoto: gestiona la conexión de usuarios remotos a procesos servidores de Windows NT.

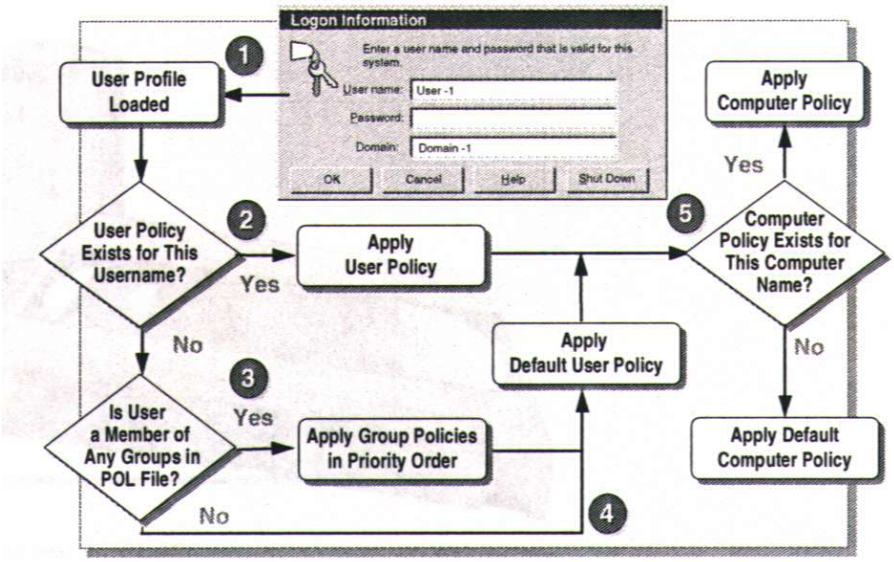

Figura 2. Diagrama de Flujo del Proceso de Inicio de Windows NT.

- **El Subsistema de Seguridad**

Este subsistema interacciona con el proceso de inicio y el llamado monitor de referencias de seguridad (se tratara en el Executive), y de esta forma se construye el modelo de seguridad en Windows NT.

El subsistema de seguridad interacciona con el proceso de inicio, atendiendo las peticiones de acceso al sistema. Consta de dos subcomponentes:

- La autoridad de seguridad local: es el corazón del subsistema. En general gestiona la política de seguridad local; así, se encarga de generar los permisos de acceso, de comprobar que el usuario que solicita conexión tiene acceso al sistema, de verificar todos los accesos sobre los objetos (para lo cual

se ayuda del monitor de referencias a seguridad) y de controlar la política de auditorías, llevando la cuenta de los mensajes de auditoría generados por el monitor de referencias. Las auditorías son una facilidad que proporciona Windows NT para monitorizar diversos acontecimientos del sistema por parte del Administrador.

- El administrador de cuentas: mantiene una base de datos con las cuentas de todos los usuarios (login, claves, identificaciones, etc.). Proporciona los servicios de validación de usuarios requeridos por el subcomponente anterior.

- **Un Subsistema de Entorno:** da soporte a aplicaciones procedentes de SO distintos, adaptándolas para su ejecución bajo Windows NT. Existen tres de este tipo:

  - **El Subsistema Win32**

Es el más importante, ya que atiende no sólo a las aplicaciones nativas de Windows NT, sino que para aquellos programas no Win32, reconoce su tipo y los lanza hacia el subsistema correspondiente. En el caso de que la aplicación sea MS−DOS o Windows de 16 bits (Windows 3.11 e inferiores), lo que hace es crear un nuevo subsistema protegido pero no servidor. Así, la aplicación DOS o Win16 se ejecutaría en el contexto de un proceso llamado VDM (Virtual DOS Machine, máquina virtual DOS), que no es más que un simulador de un ordenador funcionando bajo MS−DOS. Las llamadas al API Win16 serían correspondidas con las homónimas en API Win32. Microsoft llama a esto WOW (Windows On Win32).

El subsistema soporta una buena parte del API Win32. Así, se encarga de todo lo relacionado con la interfaz gráfica con el usuario (GUI), controlando las entradas del usuario y salidas de la aplicación. Por ejemplo, un buen número de funciones de las bibliotecas USER32 y GDI32 son atendidas por Win32, ayudándose del Executive cuando es necesario.

El funcionamiento como servidor de Win32 lo veremos un poco más adelante, en el apartado de llamadas a procedimientos locales.

  - **El Subsistema POSIX**

La norma POSIX (Portable Operating System Interface for Unix) fue elaborada por IEEE para conseguir la portabilidad de las aplicaciones entre distintos entornos UNIX. La norma se ha implementado no sólo en muchas versiones de UNIX, sino también en otros SO como Windows NT, VMS, etc. Se trata de un conjunto de 23 normas, identificadas como IEEE 1003.0 a IEEE 1003.22, o también POSIX.0 a POSIX.22, de las cuales el subsistema POSIX soporta la POSIX.1, que define un conjunto de llamadas al sistema en lenguaje C.

El subsistema sirve las llamadas interaccionando con el Executive. Se encarga también de definir aspectos específicos del SO UNIX, como pueden ser las relaciones jerárquicas entre procesos padres e hijos (las cuales no existen en el subsistema Win32, por ejemplo, y que por consiguiente no aparecen implementadas directamente en el Executive).

  - **El Subsistema OS/2**

Igual que el subsistema POSIX proporciona un entorno para aplicaciones UNIX, este subsistema da soporte a las aplicaciones OS/2. Proporciona la interfaz gráfica y las llamadas al sistema; las llamadas son servidas con ayuda del Executive.

- **El Executive**

No se debe confundir el Executive con el núcleo de Windows NT, aunque muchas veces se usan

(incorrectamente) como sinónimos. El Executive consta de una serie de componentes software, que **se ejecutan en modo privilegiado**, y uno de los cuales es el núcleo. Dichos componentes son totalmente independientes entre sí, y se comunican a través de interfaces bien definidas. Recordemos que en el diseño se procuró dejar el núcleo tan pequeño como fuera posible, y, como veremos, la funcionalidad del núcleo es mínima. Pasemos a comentar cada módulo.

1. El Administrador de Objetos (Object Manager)

Se encarga de crear, destruir y gestionar todos los objetos del Executive. Tenemos infinidad de objetos: procesos, subprocesos, ficheros, segmentos de memoria compartida, semáforos, mutex, sucesos, etc. Los subsistemas de entorno (Win32, OS/2 y POSIX) también tienen sus propios objetos. Por ejemplo, un objeto ventana es creado (con ayuda del administrador de objetos) y gestionado por el subsistema Win32. La razón de no incluir la gestión de ese objeto en el Executive es que una ventana sólo es innata de las aplicaciones Windows, y no de las aplicaciones UNIX o OS/2. Por tanto, el Executive no se encarga de administrar los objetos relacionados con el entorno de cada SO concreto, sino de los objetos comunes a los tres.

2. El Administrador de Procesos (Process Manager)

Se encarga (en colaboración con el administrador e objetos) de crear, destruir y gestionar los procesos y subprocesos. Una de sus funciones es la de repartir el tiempo de CPU entre los distintos subprocesos (ver el capítulo de los procesos). Suministra sólo las relaciones más básicas entre procesos y subprocesos, dejando el resto de las interrelaciones entre ellos a cada subsistema protegido concreto. Por ejemplo, en el entorno POSIX existe una relación filial entre los procesos que no existe en Win32, de manera que se constituye una jerarquía de procesos. Como esto sólo es específico de ese subsistema, el administrador de objetos no se entromete en ese trabajo y lo deja en manos del subsistema.

3. El Administrador de Memoria Virtual (Virtual Memory Manager)

Windows NT y UNIX implementan un direccionamiento lineal de 32 bits y memoria virtual paginada bajo demanda. El VMM se encarga de todo lo relacionado con la política de gestión de la memoria: determina los conjuntos de trabajo de cada proceso, mantiene un conjunto de páginas libres, elige páginas víctima, sube y baja páginas entre la memoria RAM y el archivo de intercambio en disco, etc. Una explicación detallada la dejaremos para el capítulo de la memoria.

4. Facilidad de Llamada a Procedimiento Local (LPC Facility)

Este módulo se encarga de recibir y envíar las llamadas a procedimiento local entre las aplicaciones cliente y los subsistemas servidores.

5. Administrador de Entrada/Salida (I/O Manager)

Consiste en una serie de subcomponentes, que son:

- El administrador del sistema de ficheros
- El servidor y el redirector de red
- Los drivers de dispositivo del sistema
- El administrador de caches

Buena parte de su trabajo es la gestión de la comunicación entre los distintos drivers de dispositivo, para lo cual implementa una interfaz bien definida que permite el tratamiento de todos los drivers de una manera homogénea, sin que intervenga el cómo funciona específicamente cada uno.

Trabaja en conjunción con otros componentes del Executive, sobre todo con el VMM. Le proporciona la E/S síncrona y asíncrona, la E/S a archivos asignados en memoria y las caches de los ficheros.

El administrador de caches no se limita a gestionar unos cuantos buffers de tamaño fijo para cada fichero abierto, sino que es capaz de estudiar las estadísticas sobre la carga del sistema y variar dinámicamente esos tamaños de acuerdo con la carga. El VMM realiza algo parecido en su trabajo, como veremos en su momento.

6. Monitor de Referencias a Seguridad

Este componente da soporte en modo privilegiado al subsistema de seguridad, con el que interacciona. Su misión es actuar de alguna manera como supervisor de accesos, ya que comprueba si un proceso determinado tiene permisos para acceder a un objeto determinado, y monitoriza sus acciones sobre dicho objeto.

De esta manera es capaz de generar los mensajes de auditorías. Soporta las validaciones de acceso que realiza el subsistema de seguridad local.

En UNIX, de la seguridad se encargaba un módulo llamado el Kerberos (Cancerbero), desarrollado por el MIT como parte del Proyecto Atenas. Kerberos se ha convertido en una norma de facto, y se incorporará a Windows NT en su versión 5.0.

7. El Núcleo (Kernel)

Situado en el corazón de Windows NT, se trata de un micro−kernel que se encarga de las funciones más básicas de todo el SO:

- Ejecución de subprocesos
- Sincronización multiprocesador
- Manejo de las interrupciones hardware

8. Nivel de Abstracción de Hardware (HAL)

Es una capa de software incluida en el Executive que sirve de interfaz entre los distintos drivers de dispositivo y el resto del sistema operativo. Con HAL, los dispositivos se presentan al SO como un conjunto homogéneo, a través de un conjunto de funciones bien definidas. Estas funciones son llamadas tanto desde el SO como desde los propios drivers. Permite a los drivers de dispositivo adaptarse a distintas arquitecturas de E/S sin tener que ser modificados en gran medida. Además oculta los detalles hardware que conlleva el multiprocesamiento simétrico de los niveles superiores del SO.

- **Llamadas a Procedimientos Locales y Remotos**

Windows NT, al tener una arquitectura cliente−servidor, implementa el mecanismo de llamada a procedimiento remoto (RPC) como medio de comunicación entre procesos clientes y servidores, situados ambos en máquinas distintas de la misma red. Para clientes y servidores dentro de la misma máquina, la RPC toma la forma de llamada a procedimiento local (LPC). Vamos a estudiar en detalle ambos mecanismos pues constituyen un aspecto fundamental del diseño de Windows NT.

1. RPC (Remote Procedure Call)

Se puede decir que el sueño de los diseñadores de Windows NT es que algún día se convierta en un **sistema distribuido puro**, es decir, que cualquiera de sus componentes pueda residir en máquinas distintas, siendo el

kernel en cada máquina el coordinador general de mensajes entre los distintos componentes. En la última versión de Windows NT esto no es aún posible.

No obstante, el mecanismo de RPC permite a un proceso cliente acceder a una función situada en el espacio virtual de direcciones de otro proceso servidor situado en otra máquina de una manera totalmente transparente.

Vamos a explicar el proceso en conjunto. Supongamos que se tiene un proceso cliente ejecutándose bajo una máquina A, y un proceso servidor bajo una máquina B. El cliente llama a una función f de una biblioteca determinada. El código de f en su biblioteca es una *versión especial* del código real; el código real reside en el espacio de direcciones del servidor. Esa versión especial de la función f que posee el cliente se denomina **proxy**. El código proxy lo único que hace es recoger los parámetros de la llamada a f, construye con ellos un mensaje, y pasa dicho mensaje al Executive. El Executive analiza el mensaje, determina que va destinado a la máquina B, y se lo envía a través del interfaz de transporte. El Executive de la máquina B recibe el mensaje, determina a qué servidor va dirigido, y llama a un código especial de dicho servidor, denominado **stub**, al cual le pasa el mensaje. El stub desempaqueta el mensaje y llama a la función f con los parámetros adecuados, ya en el contexto del proceso servidor. Cuando f retorna, devuelve el control al código stub, que empaqueta todos los parámetros de salida (si los hay), forma así un mensaje y se lo pasa al Executive.

Ahora se repite el proceso inverso; el Executive de B envía el mensaje al Executive de A, y este reenvía el mensaje al proxy. El proxy desempaqueta el mensaje y devuelve al cliente los parámetros de retorno de f. Por tanto, para el cliente todo el mecanismo ha sido transparente. Ha hecho una llamada a f, y ha obtenido unos resultados; ni siquiera tiene que saber si el código real de f está en su biblioteca o se encuentra en una máquina situada tres plantas más abajo.

2. LPC (Local Procedure Call)

Las LPC se pueden considerar una versión *descafeinada* de las RPC. Se usan cuando un proceso necesita los servicios de algún subsistema protegido, típicamente Win32. Se intentara descubrir su funcionamiento.

El proceso cliente tiene un espacio virtual de 4 Gb. Los 2 Gb inferiores son para su uso (excepto 128 Kb). Los 2 Gb superiores son para uso del sistema.

Vamos a suponer que el cliente realiza una llamada a la función CreateWindow. Dicha función crea un objeto ventana y devuelve un descriptor al mismo. No es gestionada directamente por el Executive, sino por el subsistema Win32 (con algo de colaboración por parte del Executive, por supuesto; por ejemplo, para crear el objeto). El subsistema Win32 va guardando en su propio espacio de direcciones una lista con todos los objetos ventana que le van pidiendo los procesos. Por consiguiente, los procesos no tienen acceso a la memoria donde están los objetos; simplemente obtienen un descriptor para trabajar con ellos. Cuando el cliente llama a CreateWindow, se salta al código de esa función que reside en la biblioteca USER32.DLL asignada en el espacio de direcciones del cliente.

Por supuesto, ese no es el código real, sino el proxy. El proxy empaqueta los parámetros de la llamada, los coloca en una zona de memoria compartida entre el cliente y Win32, pone al cliente a dormir y ejecuta una LPC. La facilidad de llamada a procedimiento local del Executive captura esa llamada, y en el subsistema Win32 se crea un subproceso que va a atender a la petición del cliente. Ese subproceso es entonces despertado, y comienza a ejecutar el correspondiente código de stub. Los códigos de stub de los subsistemas se encuentran en los 2 Gb superiores (los reservados) del espacio virtual del proceso cliente. Aunque no he encontrado más documentación al respecto, es muy probable que dichos 2 Gb sean los mismos que se ven desde el espacio virtual de Win32. Sea como sea, el caso es que el stub correspondiente desempaqueta los parámetros del área de memoria compartida y se los pasa a la función CreateWindow situada en el espacio de Win32. Ése sí es el código real de la función. Cuando la función retorna, el stub continúa, coloca el descriptor

a la ventana en la memoria compartida, y devuelve el control de la LPC al Executive. El subproceso del Win32 es puesto a dormir. El Executive despierta al subproceso cliente, que estaba ejecutando código proxy. El resto de ese código lo que hace es simplemente tomar el descriptor y devolverlo como resultado de la función CreateWindow.

III. PROCESOS

- **Definición de Proceso y Sub Proceso**

Debemos tener cuidado con no confundir el proceso en Windows NT con el proceso en los SO más clásicos, como UNIX. Vamos a intentar dar una definición general de lo que entiende Windows NT por proceso y subproceso, aunque después iremos perfilando poco a poco ambos conceptos.

Un **proceso** es una entidad *no ejecutable* que posee un espacio de direcciones propio y aislado, una serie de recursos y una serie de subprocesos. En el espacio de direcciones hay colocado algún código ejecutable (entre otras cosas). Bien, hemos dicho que un proceso es una entidad "no−ejecutable". En efecto, no puede ejecutar el código de su propio espacio de direcciones, sino que para esto le hace falta al menos un subproceso. Por consiguiente, un **subproceso** es la *unidad de ejecución* de código. Un subproceso está asociado con una serie de instrucciones, unos registros, una pila y una cola de entrada de mensajes (enviados por otros procesos o por el SO).

Cuando se crea un proceso, automáticamente se crea un subproceso asociado (llamado **subproceso primario**). Los subprocesos también se llaman "hebras de ejecución" (threads of execution). Debe quedarnos muy claro, pues, que lo que se ejecutan son subprocesos, no procesos. Los procesos son como el soporte sobre el que corren los subprocesos. Y entre los subprocesos se reparte el tiempo de CPU.

Podemos pensar en los subprocesos de Windows NT como los procesos de los SO clásicos (aunque existen matices, como sabemos). A veces, por comodidad y por costumbre, usaremos ambos términos como sinónimos, y diremos que "el proceso A llama a CreateWindow", aunque se debe entender que "un subproceso del proceso A llama a CreateWindow".

Un proceso tiene un espacio de direcciones virtuales de 4 Gb. En algún lugar de ese espacio se halla un código ejecutable (que quizás es la imagen de un programa en disco). Un proceso puede crear subprocesos, estando su número fijado por el sistema. Se dice que muere cuando *todos* sus subprocesos han muerto (incluso aunque el subproceso primario haya muerto, si aún existe algún subproceso propiedad del proceso, el proceso seguirá vivo).

- **Planificación del Tiempo de la CPU por Round Robin con Prioridades**

Windows NT utiliza la planificación del anillo circular o round robin. Esta técnica consiste en que los subprocesos que pueden ser ejecutados se organizan formando un anillo, y la CPU va dedicándose a cada uno durante un tiempo. El tiempo máximo que la CPU va a estar dedicada a cada uno se denomina quantum, y es fijado por el Administrador del Sistema.

Si el subproceso está esperando por alguna entrada−salida o por algún suceso, la CPU lo pondrá a dormir, y pondrá en ejecución al siguiente del anillo. Si un subproceso que la CPU está ejecutando consume su quantum, la CPU también lo pondrá a dormir, pasando al siguiente.

En Windows NT, existe un rango de prioridades que va del 1 al 31, siendo 31 la más alta. Todo proceso y subproceso tienen un valor de prioridad asociado.

Existe un anillo o cola circular por cada uno de los niveles de prioridad. En cada anillo están los subprocesos

de la misma prioridad. El Executive comienza a repartir el tiempo de CPU en el primer anillo de mayor prioridad no vacío. A cada uno de esos subprocesos se les asigna secuencialmente la CPU durante el tiempo de un quantum, como ya indicamos antes. Cuando todos los subprocesos de nivel de prioridad n están dormidos, el Executive comienza a ejecutar los del nivel -(n−1), siguiendo el mismo mecanismo.

Análogamente, si un subproceso se está ejecutando, y llegara uno nuevo de prioridad superior, el Executive suspendería al primero (aunque no haya agotado su quantum), y comenzaría a ejecutar el segundo (asignándole un quantum completo).

1. Prioridad de proceso y subproceso

Un proceso se dice que pertenece a una clase de prioridad. Existen cuatro clases de prioridad, que son:

- Desocupado. Corresponde a un valor de prioridad 4.
- Normal. Corresponde a un valor de prioridad 7 ó 9.
- Alta. Corresponde a un valor de prioridad 13.
- Tiempo Real. Corresponde a un valor de prioridad 24.

La clase "Normal" es la que el Executive asocia a los procesos por defecto. Los procesos en esta clase se dice que tienen una prioridad dinámica: el Executive les asigna un valor de 7 si se están ejecutando en segundo plano, mientras que si pasan a primer plano, la prioridad se les aumenta a un valor de 9.

La clase "Desocupado" va bien para procesos que se ejecuten periódicamente y que por ejemplo realicen alguna función de monitorización.

La clase "Alta" la tienen procesos tales como el Administrador de Tareas (Task Manager). Dicho proceso está la mayor parte del tiempo durmiendo, y sólo se activa si el usuario pulsa Control−Esc. Entonces, el SO inmediatamente pone a dormir al subproceso en ejecución (aunque no haya agotado su quantum) y ejecuta al subproceso correspondiente del proceso Administrador de Tareas, que visualizará el cuadro de diálogo característico, mostrándonos las tareas actuales.

La clase "Tiempo Real" no es recomendable que la tenga ningún proceso normal. Es una prioridad más alta incluso que muchos procesos del sistema, como los que controlan el ratón, el teclado, el almacenamiento en disco en segundo plano, etc. Es evidente que usar esta prioridad sin un control extremo puede causar consecuencias nefastas.

Así como un proceso tiene una prioridad oscilando entre cuatro clases, un subproceso puede tener cualquier valor en el rango [1,31]. En principio, cuando el subproceso es creado, su prioridad es la correspondiente a la de la clase de su proceso padre. Pero este valor puede ser modificado si el subproceso llama a la función

BOOL SetThreadPriority (HANDLE hThread, int nPriority);

Donde:

hThread es el descriptor del subproceso

nPriority puede ser:

- THREAD_PRIORITY_LOWEST : resta 2 a la prioridad del padre
- THREAD_PRIORITY_BELOW_NORMAL: resta 1 a la prioridad del padre

- THREAD_PRIORITY_NORMAL: mantiene la prioridad del padre
- THREAD_PRIORITY_ABOVE_NORMAL: suma 1 a la prioridad del padre
- THREAD_PRIORITY_HIGHEST: suma 2 a la prioridad del padre
- THREAD_PRIORITY_IDLE: hace la prioridad igual a 1, independientemente de la prioridad del padre
- THREAD_PRIORITY_TIME_CRITICAL: hace la prioridad igual a 15 si la clase de prioridad del padre es desocupada, normal o alta; si es tiempo real, entonces hace la prioridad igual a 31

De esta manera es como calcula el Executive la prioridad de un subproceso. Por tanto, la prioridad de un subproceso es relativa a la de su padre (excepto en IDLE y TIME_CRITICAL). Mediante suma y resta de la prioridad del padre obtenemos todo el rango de prioridades:

Clase proceso padre **Prior. Subproceso**	**Clase de-socupado**	**Clase normal en primer plano**	**Clase normal en segundo plano**	**Clase alta**	**Clase tiempo real**
Crítico en tiempo	15,00	15,00	15,00	15,00	31,00
Más alta	6,00	9,00	11,00	15,00	26,00
Más que normal	5,00	8,00	10,00	14,00	25,00
Normal	4,00	7,00	9,00	13,00	24,00
Menos que normal	3,00	6,00	8,00	12,00	23,00
Más baja	2,00	5,00	7,00	11,00	22,00
Desocupado	1,00	1,00	1,00	1,00	16,00

La ventaja de este sistema de las prioridades relativas es que si un proceso cambia su clase de prioridad durante su vida, sus subprocesos hijos tendrían sus prioridades automáticamente actualizadas.

2. Creación y destrucción de procesos

La llamada al sistema que crea un proceso es una de las más complejas de todo el API Win32. Vamos a comentarla para comprender mejor la forma en la que el Executive trabaja.

Un proceso se crea cuando un subproceso de otro proceso hace una llamada a:

BOOL CreateProcess (LPCTSTR lpszImageName, LPCTSTR lpszCommandLine, LPSECURITY_ATTRIBUTES lpsaProcess, LPSECURITY_ATTRIBUTES lpsaThread, BOOL fInheritHandles, DWORD fdwCreate, LPVOID lpvEnvironment, LPTSTR lpszCurDir, LPSTARTUPINFO lpsiStartInfo, LPROCESS_INFORMATION lppiProcInfo);

El Executive crea un espacio virtual de 4 Gb para el proceso, y también crea el subproceso primario. Veamos el significado de los parámetros:

lpszImageName: es el nombre del archivo que contiene el código ejecutable que el Executive asignará en el espacio virtual del proceso. Si es NULL, entonces se entenderá que viene dado en el siguiente parámetro.

lpszCommandLine: argumentos para el nombre del archivo

lpsaProcess, lpsaThread y fInheritHandles: los dos primeros son punteros con los que podemos dar unos atributos de seguridad para el proceso y su subproceso primario. Si pasamos NULL, el sistema pondrá los valores por defecto. Los parámetros son punteros a una estructura del tipo:

typedef struct _SECURITY_ATTRIBUTES {

DWORD nLength;

LPVOID lpSecurityDescriptor;

BOOL bInheritHandle;

} SECURITY_ATTRIBUTES;

El campo lpSecurityDescriptor se refiere a permisos sobre el objeto proceso, pero no he encontrado más información al respecto.

De esta estructura vamos a destacar el campo bInheritHandle, que se refiere a la herencia. En Windows NT, cualquier objeto que creemos va a tener asociada una estructura de este tipo en donde se indicará, con el parámetro bInheritHandle, si dicho objeto es heredable o no. El objeto es propiedad de un proceso. Ese proceso puede crear procesos hijos; dichos procesos, por defecto, no heredarán ninguno de los objetos de su padre. Los procesos hijos que tengan la capacidad de heredar, heredarán aquellos objetos de su padre que tengan el campo bInheritHandle a TRUE.

Ahora bien, un proceso y un subproceso son también objetos. Por tanto, ambos objetos podrán ser heredables por otros procesos. Si el campo bInheritHandle es TRUE en la estructura apuntada por lpsaProcess, entonces el proceso que estamos creando será heredable por otros procesos hijos de su mismo padre.

Si el campo bInheritHandle es TRUE en la estructura apuntada por lpsaThread, entonces e subproceso primario del proceso que estamos creando será igualmente heredable.

Resta explicar el parámetro fInheritHandles. Si vale TRUE, entonces el proceso que estamos creando podrá heredar todos los objetos heredables de su padre (es decir, aquellos objetos cuyo campo bInheritHandle sea TRUE).

No se debe confundir todo esto con herencia entre procesos y subprocesos hijos. Un subproceso siempre podrá tener acceso a los objetos creados por el proceso al que pertenece (o mejor dicho, creados por algún otro subproceso del proceso al que pertenece).

fdwCreate: es una máscara donde se pueden especificar (mediante OR lógica) muchos indicadores para el proceso a crear. Los más importantes son:

la clase de prioridad del proceso: IDLE_PRIORITY_CLASS (desocupado), NORMAL_PRIORITY_CLASS (normal), HIGH_PRIORITY_CLASS (alta), REALTIME_PRIORITY_CLASS (tiempo real)

si el proceso va a ser dormido al crearse, usando CREATE_SUSPENDED

lpvEnvironment: apunta a un bloque de memoria que contiene cadenas de entorno. Si vale NULL, usará las mismas cadenas que su proceso padre. Las cadenas de entorno no son más que variables con algún valor que usarán los procesos con algún fin, (por ejemplo, el directorio home, el path). Tiene un significado similar al concepto de entorno de UNIX.

lpszCurDir: cadena con directorio y unidad de trabajo para el nuevo proceso.

lpsiStartInfo: apunta a una estructura bastante grande que no vamos a escribir. Los campos de dicha estructura dan informaciones al subsistema Win32 como por ejemplo si el proceso va a estar basado en GUI o en consola (GUI es con ventanas; consola es simulando el modo texto), el tamaño de la ventana inicial del proceso, las coordenadas de dicha ventana, su tamaño, su título ...

lppiProcInfo: apunta a una estructura que rellena la llamada antes de devolver el control, y es de suma importancia:

typedef struct _PROCESS_INFORMATION{

HANDLE hProcess;

HANDLE hThread;

DWORD dwProcessId;

DWORD dwThreadId;

} PROCESS_INFORMATION;

En hprocess y hThread el Executive devuelve un par de descriptores a los objetos proceso y subproceso primario recién creados, y que servirán para hacer referencias a los mismos. Cada vez que el Executive crea cualquier objeto, asocia con dicho objeto un contador de uso. Cada vez que un proceso distinto usa un mismo objeto, incrementa el contador en 1. Cada vez que un proceso libera un objeto, decrementa el contador en 1. Cuando el contador de uso de un objeto es 0, el Executive destruye el objeto. Pues bien, cuando CreateProcess devuelve el control, los objetos proceso y subproceso primario tienen sus respectivos contadores con valor 2. De este modo, para que el Executive pueda destruirlos, habrán de ser liberados dos veces: una, cuando ellos mismos terminen (el contador pasaría a valer 1), y otra, cuando el proceso padre los libere (o sea, cuando cierre los descriptores; así, sus contadores valdrían 0 y 0, y el Executive los destruiría). De aquí es infiere que es vital que el proceso padre cierre esos descriptores (si no, no se destruirían los objetos y podría desbordarse la memoria). La llamada para cerrar descriptores es

BOOL CloseHandle (HANDLE hobject);

dwProcessId y dwThreadId son unos identificadores únicos que el Executive asocia al proceso y subproceso primario, respectivamente, análogos al PID en UNIX.

Hasta aquí la llamada al sistema que nos permite crear procesos. La llamada para finalizar el proceso es, afortunadamente, mucho más simple:

VOID ExitProcess (UINT fuExitCode);

que devuelve el entero fuExitCode, que es un código de salida que el proceso envía antes de finalizar (que diga si ha finalizado con éxito, si no, etc). Cuando un proceso termina, se realizan las siguientes acciones:

- Todos los subprocesos del proceso finalizan
- Se cierran todos los descriptores de objetos del Executive y de Win32 abiertos por el proceso
- El estado del objeto proceso para a estar señalado

- El estado de terminación del proceso se pone al código de salida adecuado

Hemos dicho que el objeto proceso se pone a estado señalado. Un objeto puede tener dos estados: señalado y no señalado, estados que utiliza el sistema y los propios procesos para varias funciones. El curioso lector puede consultar el apartado de "Comunicación entre procesos".

3. Creación y destrucción de subprocesos

Análogamente a como nos hemos auxiliado de la llamada al sistema CreateProcess para comprender el mecanismo de creación de procesos, vamos a hacer lo propio para la creación de subprocesos. Un subproceso se crea cuando otro subproceso hace una llamada a:

HANDLE CreateThread (LPSECURITY_ATTRIBUTES lpsa, DWORD cbStack, LPTHREAD_START_ROUTINE lpStartAddr, LPVOID lpvThreadParm, DWORD fdwCreate, LPDWORD lpIDThread);

lpsa: tiene el mismo significado que en CreateProcess, es decir, un puntero a una estructura SECURITY_ATTRIBUTES, donde se especifican permisos del subproceso y si es heredable o no.

cbStack: vamos a aprovechar este parámetro para explicar la pila de un subproceso. Todo subproceso tiene una pila asociada situada en el espacio virtual del proceso al que pertenece. Virtualmente, la pila tiene un tamaño por defecto de 1 Mb (y además es el máximo; tamaños inferiores pueden ser indicados al enlazar la aplicación). De esa pila virtual el subproceso puede tener asignada en memoria un trocito. El tamaño de ese trocito viene dado por el parámetro cbStack, y por defecto es 1 página (4 Kb). Técnicamente, se dice que la pila tiene reservado un espacio de 1 Mb, y asignado un espacio de cbStack bytes. (el significado de ambos términos lo veremos detenidamente en el capítulo de administración de la memoria). Si el subproceso desborda su trocito de pila asignado en memoria se eleva una excepción; el Executive captura la excepción y asigna otros cbStack bytes en memoria física para la pila del subproceso. Por tanto, la pila crece dinámicamente en trozos de cbStack bytes. Lo más eficiente es que cbStack sea múltiplo del tamaño de la página.

lpStartAddr, lpThreadParm: ya dijimos que todo subproceso ejecuta una porción de código del proceso al que pertenece. Este parámetro apunta a la función que contiene el código a ejecutar por nuestro subproceso. Es posible hacer que varios subprocesos tengan la misma función asociada. El perfil de la función es fijo:

```
DWORD WINAPI NombreFuncionSubproceso (LPVOID lpvThreadParm)

{

DWORD dwresult =0;

...

return (dwresult);

}
```

El parámetro lpvThreadParm de la función es justamente el mismo que se le pasa a CreateThread; de hecho, el sistema se limita a pasar dicho parámetro a la función asociada al subproceso cuándo éste comienza su ejecución.

El parámetro puede ser un valor de 32 bits o un puntero de 32 bits. Puede servir para dar algún tipo de

inicialización al subproceso.

El parámetro dwresult de la anterior función es un código de salida que el subproceso devuelve cuando acaba de ejecutar código. Es similar al código que devuelven los procesos al acabar.

fdwCreate: si vale 0, entonces el subproceso comienza a ejecutarse en cuanto esté creado. Si vale CREATE_SUSPENDED, entonces se creará, pero automáticamente será suspendido.

lpIDThread: debe ser un puntero a una estructura DWORD, donde el Executive pondrá el identificador que le ha asignado al subproceso. Es obligatorio pasar una dirección válida.

El subproceso recién creado iniciará su ejecución inmediatamente antes del retorno de CreateThread (a menos que hayamos especificado el indicador CREATE_SUSPENDED).

Un subproceso se puede autodestruir llamando a:

VOID ExitThread (DWORD fdwExitCode);

Cualquier código situado a continuación de esa llamada no será ejecutado.

fdwExitCode es el código de salida del subproceso.

Un subproceso puede destruir a otro llamando a:

BOOL TerminateThread (HANDLE hThread, DWORD dwExitCode);

que mata al subproceso cuyo descriptor es hThread, y devuelve su código de salida en dwExitCode.

Si un subproceso termina por él mismo, el Executive destruye su pila asociada, pero si termina porque algún otro lo ha matado, entonces no la destruye hasta que no finalice su padre (pues otros subprocesos pueden estar haciendo uso de objetos alojados en dicha pila). Podemos resumir las acciones relacionadas con la muerte de un subproceso:

- Se cierran todos los descriptores de objetos del subsistema Win32 que el subproceso ha estado usando.
- El estado del objeto subproceso pasa a señalado.
- Su estado de terminación toma el código de salida.
- Si es el último subproceso vivo del proceso padre, éste también muere.

- **Comunicación y Sincronización de Procesos Mediante Objetos**

En Windows NT, los mecanismos clásicos de sincronización entre procesos (como los semáforos, las regiones críticas, los sucesos, etc.) son tratados como objetos. Es más, existen objetos no específicos de sincronización pero que también pueden ser usados con estos fines. Por tanto, vamos en primer lugar a enumerar todos los objetos de sincronización y a dar algunas características globales, para posteriormente adentrarnos a estudiar los más importantes.

Podemos sincronizar subprocesos mediante alguno de los siguientes objetos:

- Semáforos
- Mutexes
- Sucesos

- Archivos
- Procesos
- Subprocesos
- Entrada del terminal
- Notificación de cambio de archivo

Antes hemos mencionado las regiones críticas. Aunque Windows NT las incluye como mecanismo de sincronización, no las trata explícitamente como objeto. No obstante también las estudiaremos en este apartado por mantener la homogeneidad.

Como ya esbozamos en el capítulo de los procesos, en cualquier instante un objeto está en uno de dos estados: señalado (1) y no señalado (0). Cada estado tiene un significado dependiendo de la clase del objeto.

Por ejemplo, en el apartado anterior vimos que durante la vida de un proceso o un subproceso su estado era no señalado, pero que al morir pasaban al estado señalado. De aquí que ambos tipos de objetos sirvan para la sincronización. Por ejemplo, un subproceso A puede querer dormir hasta que otro proceso/subproceso B acabe; por tanto, A dormirá mientras el objeto asociado al B esté no señalado. En cuanto B pase a señalado, A despertará.

Igualmente, un subproceso se puede sincronizar con el fin de una lectura/escritura en un archivo. En general, cuando alguna de estas operaciones finalizan, el objeto archivo en cuestión pasa a estado señalado.

El objeto asociado a la entrada de teclado se pone señalado cuando existe algo en el buffer de entrada. La aplicación de este hecho para sincronización es evidente. Un subproceso puede así estar durmiendo mientras el buffer esté vacío.

El resto de objetos los veremos con más detalle a lo largo de este capítulo. Pero antes vamos a dar las llamadas al sistema que se utilizan para la sincronización con objetos.

Se trata de:

DWORD WaitForSingleObject (HANDLE hObject, DWORD dwTimeout);

Esta llamada simplemente mantiene al subproceso que la realiza dormido hasta que el objeto identificado por hObject pase al estado señalado. El parámetro dwTimeout indica el tiempo (en ms) que el subproceso quiere esperar por el objeto. Si vale 0, entonces la llamada sólo hace una comprobación del estado y retorna inmediatamente; si devuelve WAIT_OBJECT_0, el objeto estaba señalado; si devuelve WAIT_TIMEOUT, el objeto estaba no señalado. Si como tiempo metemos INFINITE, el subproceso dormirá hasta que el objeto pase a señalado. Hay un par de códigos de salida más que comentaremos en su momento (cuando expliquemos los mutex).

DWORD WaitForMultipleObjects (DWORD cObjects, LPHANDLE lpHandles, BOOL bWaitAll, DWORD dwTimeout);

Es parecida a la anterior pero da la posibilidad de esperar por varios objetos o bien por alguno de una lista de objetos. cObjects indica el número de objetos a comprobar. lpHandles apunta a una matriz que contiene descriptores a los objetos. El booleano bWaitAll indica si queremos esperar a que todos los objetos pasen a estado señalado o tan sólo uno, y dwTimeout es el tiempo a esperar. Si hay varios objetos que han pasado al estado señalado, la llamada coge sus descriptores, toma el menor y devuelve su posición dentro de la matriz (sumada a un código de retorno análogo a los de WaitForSingleObject).

El tema de sincronización está íntimamente relacionado con el de interbloqueos. Supongamos que tenemos

dos subprocesos A y B compitiendo por dos objetos 1 y 2, esperando a que ambos estén señalados, para lo cual han hecho sendas llamadas a WaitForMultipleObjects. Supongamos que 1 pasa a estado señalado. Y supongamos que el Executive decidiera otorgar ese hecho al proceso A, colocando a 1 de nuevo a no señalado. Mientras tanto, 2 pasa también a señalado, y el Executive otorga este hecho a B, y también pone a 2 a no señalado. En esta situación, ninguno de los dos subprocesos podría terminar nunca, pues cada uno estaría esperando a que el objeto del otro pasara a señalado. Entonces A y B están interbloqueados. Para evitar esta situación, el Executive no entrega los objetos hasta que ambos estén señalados; en ese momento despertaría a uno de los subprocesos. El otro seguiría dormido hasta que el primer subproceso terminara de trabajar con los objetos.

A continuación se estudia cada uno de los objetos específicos de sincronización de Windows NT.

1. Secciones Críticas

Las secciones críticas son un mecanismo para sincronizar subprocesos pertenecientes a un mismo proceso, pero, como ya hemos indicado, no son objetos.

Una sección o región crítica (SC) es un trozo de código ejecutable tal que para que un subproceso pueda ejecutarlo se requiere que tenga acceso a una estructura de datos especial y compartida.

Dicha estructura es del tipo CRITICAL_SECTION, cuyos campos no son interesantes, y además no son accesibles directamente, sino a través de una llamada al subsistema Win32:

VOID InitializeCriticalSection (LPCRITICAL_SECTION lpCriticalSection);

Veamos algunas llamadas para manejar las SC:

VOID EnterCriticalSection (LPCRITICAL_SECTION lpCriticalSection);

VOID LeaveCriticalSection (LPCRITICAL_SECTION lpCriticalSection);

La primera sirve para que un subproceso solicite entrar en una SC. La segunda permite salir de la SC a un subproceso que esté dentro de la misma.

La función de entrar opera así: la primera vez que es llamada, se registra en la estructura CRITICAL_SECTION un identificador del subproceso que la posee.

Si antes de que el subproceso la abandone, el SO entrega la CPU a otro (el caso más frecuente), y entonces ese otro subproceso hace la llamada para entrar en la SC, entonces la función ve que la estructura ya está en uso, con lo que el segundo subproceso sería puesto a dormir.

Cuando la CPU vuelva al primero y éste haga una llamada para salir, la estructura será asignada al segundo subproceso.

Si un subproceso vuelve a llamar a la función de entrar estando ya dentro de la SC, simplemente se incrementará un contador de uso asociado con el objeto SC. Más tarde, deberá hacer tantas otras llamadas a la función de salir para poner el contador a 0; en caso contrario, ningún otro subproceso podría ocupar la SC.

La estructura CRITICAL_SECTION y todos los recursos que el SO le hubiera asignado se pueden eliminar haciendo una llamada a

VOID DeleteCriticalSection (LPCRITICAL_SECTION lpCriticalSection);

Sería catastrófico usar esta función estando un subproceso dentro.

2. Exclusión Mutua (Mutex)

Los objetos exclusión mutua (abreviadamente mutex, de mutual exclusión) son muy parecidos a las SC, pero permiten sincronizar subprocesos pertenecientes a procesos distintos.

Se crean con la llamada:

HANDLE CreateMutex (LPSECURITY_ATTRIBUTES lpsa, BOOL fInitialOwner, LPTSRT lpszMutexName);

fInitialOwner: dice si el subproceso que crea el mutex ha de ser su propietario inicial o no. Si vale TRUE, entonces el estado inicial del objeto mutex sería no señalado, por lo que todo subproceso que espere por él sería inmediatamente puesto a dormir. Si vale FALSE, el mutex se crea con estado señalado, por lo que al primer proceso que estuviera esperando le sería asignado y podría continuar ejecutándose.

lpszMutexName: apunta a una cadena con el nombre que le hemos querido dar al objeto (o NULL si no se pone un nombre).

La llamada devuelve un descriptor al objeto creado.

Si otro subproceso llamara a la función pasándole el mismo nombre de mutex, el SO comprobaría que ya está creado y devolvería otro descriptor al mismo objeto.

HANDLE OpenMutex (DWORD fwdAccess, BOOL fInherit, LPTSTR lpszMutexName);

Esta función comprobaría si existe algún objeto mutex con nombre lpszMutexName. Si es así, devolvería un descriptor al mismo. Si no, devolvería NULL. El booleano fInherit permite que el mutex sea heredado por los subprocesos hijos.

Si el mutex no tuviera nombre, un subproceso podría obtener un descriptor al mismo llamando a DuplicateHandle.

Otra diferencia de los mutex con las SC (y en general con cualquier objeto de sincronización en Windows NT) es que un subproceso mantiene la propiedad de un mutex hasta que quiera liberarlo, pero hasta el punto de que, si el subproceso muere y no ha liberado al mutex, éste seguiría siendo de su propiedad.

Así, si un mutex está libre (señalado) y es tomado por un subproceso (pasa a no señalado), y dicho subproceso finaliza antes de liberarlo, el estado del mutex pasaría a señalado; los subprocesos que estuvieran esperando por el mutex serían despertados pero no se les asignaría el objeto a ninguno, sino que con el valor WAIT_ABANDONED de retorno de las llamadas WaitFor...Object(s) se les informaría de lo que ha sucedido, de que el mutex no ha sido liberado sino abandonado. Esto se considera como una situación de fallo en un programa.

Para liberar un mutex usaremos la llamada

BOOL ReleaseMutex (HANDLE hMutex);

Donde:

hMutex: es un descriptor al objeto. La función decrementa el contador de uso que el subproceso tiene sobre el

mutex. Cuando sea 0, el objeto podrá ser asignado al primer subproceso que por él esté esperando, igual que con las SC.

3. Semáforos

Un semáforo es un objeto distinto de las SC y los mutex. A diferencia de ambos, el objeto semáforo puede ser poseído a la vez por varios subprocesos, y no posee dentro ningún identificador del subproceso/s que lo está usando. Lo que tiene es un contador de recursos, que indica el número de subprocesos que pueden acceder al semáforo. Cuando un subproceso toma el objeto semáforo, el SO mira si el contador de recursos del mismo es 0. De ser así, pone al subproceso a dormir. Si no es 0, asigna el semáforo al subproceso y decrementa el contador. Cada vez que un subproceso libera el semáforo, se incrementa el contador.

Un semáforo está señalado cuando su contador es mayor que 0, y no señalado cuando su contador vale 0.

Un semáforo se crea con la llamada:

HANDLE CreateSemaphore (LPSECURITY_ATTIBUTES lpsa, LONG cSemInitial, LONG cSemMax, LPTSTR lpszSemName);

cSemInitial es el valor inicial no negativo del contador de recursos.

cSemMax es el valor máximo que puede alcanzar dicho contador (por tanto 0 <= cSemInitial <= cSemMax)

lpszSem-Name es el nombre que le damos al objeto.

HANDLE OpenSemaphore (DWORD fdwAccess, BOOL fInherit, LPTSTR lpszName);

La semántica de esta llamada es análoga a la de OpenMutex.

Para incrementar el contador de referencia del semáforo se usa:

HANDLE ReleaseSemaphore (HANDLE hSemaphore, LONG cRelease, LPLONG lplPrevious);

Donde:

cRelease indica el número de veces que queremos incrementar el contador (el número de veces que liberamos el semáforo). A la vuelta de la función, tendremos en lplPrevious un puntero al valor anterior del contador. Por tanto, si queremos averiguar el valor del contador tenemos que modificarlo. Ni siquiera llamando a la función con cRelease igual a 0 podríamos saber el valor anterior sin modificar el semáforo, pues entonces la función devuelve 0 como dato apuntado por lplPrevious.

4. Sucesos.

Los sucesos son objetos utilizados para indicar a los subprocesos que ha ocurrido algo en el entorno. Se puede distinguir dos tipos de objetos suceso:

- Sucesos con inicialización manual.
- Sucesos con autoinicialización.

Cualquier objeto suceso podrá estar en estado señalado o no señalado. No señalado significa que la situación asociada con el objeto aún no ha ocurrido. Señalado indica que sí se ha producido.

Ambos tipos de objeto se crean con la misma llamada:

HANDLE CreateEvent (LPSECURITY_ATTIBUTES lpsa, VOOL fManualReset, BOOL fInitialState, LPTSTR lpszEventName);

fManualReset a TRUE le indicará al SO que queremos crear un suceso de inicialización manual, y a FALSE, un suceso de autoinicialización.

fInitialState indica el estado inicial del objeto; un valor TRUE significa que el suceso se creará como obejto señalado, y a FALSE como no señalado.

Como vimos con los otros tipos de objetos de sincronización, otros procesos pueden acceder al mismo objeto usando CreateEvent y el mismo nombre, o usando la herencia, o con DuplicateHandle, o con:

HANDLE OpenEvent (DWORD fdwAccess, BOOL fInherit, LPTSTR lpszName);

a. Sucesos con inicialización manual (manual reset)

Este tipo de objetos se usan para que, cuando el suceso ocurra, es decir, se ponga a señalado, todos los subprocesos que esperaban por ese suceso se despierten a la vez y todos puedan seguir ejecutándose. Por tanto, ahora las funciones WaitFor...Objext(s) no van a tocar el estado del objeto, sino que debe hacerlo el propio subproceso que creó el objeto. Dicho subproceso puede usar las llamadas:

BOOL SetEvent (HANDLE hEvent);

BOOL ResetEvent (HANDLE hEvent);

que ponen, respectivamente, el suceso identificado por hEvent en estado señalado y no señalado. O sea, con SetEvent indicaremos que la situación asociada al obejto se ha producido, y con ResetEvent indicaremos lo contrario.

Existe una llamada muy útil con este tipo de objetos:

BOOL PulseEvent (HANDLE hEvent);

que le da un "pulso" al suceso hEvent: lo pone en estado señalado, con lo que se libera a todos los subprocesos en espera (que se seguirán ejecutando), y entonces lo pone a no señalado, con lo que los subprocesos que a partir de ese momento esperen por el suceso serán dormidos. Por tanto, equivale a SetEvent + liberación + ResetEvent.

b. Sucesos con autoinicialización (auto−reset)

Con estos objetos las funciones WaitFor...Object(s) van a funcionar como con cualquier otro tipo de objeto. Por tanto, cuando la situación asociada con el suceso ocurra y éste se ponga a señalado (con SetEvent), sólo uno de los subprocesos que estaban esperando podrá continuar, y su función Wait-For...Object(s) correspondiente pondrá el estado del objeto a no señalado. La función ResetEvent no tendría efecto aquí. La función PulseEvent pondría el suceso a señalado, liberaría un sólo subproceso de los que están esperando, y pondría el suceso a no señalado.

IV. ADMINISTRACION DE LA MEMORIA

La parte de Windows NT que soporta la gestión de la memoria se denomina **Gestor de Máquina Virtual**

(VMM), y reside en el Executive, por encima del núcleo pero ejecutándose en modo supervisor, como vimos en la introducción.

En Windows NT se utiliza memoria virtual paginada. En algunas máquinas, como las CPU Intel 386 en adelante, combina paginación con segmentación. El tamaño de la página depende de la máquina. En la CPU anterior es de 4 Kb.

A. Espacio de Direcciones de un Proceso

Todo proceso que se crea en Windows NT posee un espacio de direcciones virtuales de 4 Gb exclusivos de él. Ningún otro podrá acceder a esas direcciones, sencillamente porque ¡no las ve!. Para un proceso, todo lo que ve es suyo, y ve virtualmente 4 Gb. Los 2 Gb superiores están reservados al SO, así como los 64 Kb primeros y los 64 Kb últimos de los 2 Gb inferiores. El resto de los 2 Gb inferiores son para uso del proceso. Distinguimos varias zonas:

Direcciones para las DLL del sistema (NTDLL, KERNEL32, USER32, GDI32 ...).

Direcciones para las DLL propias de la aplicación.

Bloques y pilas de los subprocesos.

Imagen del archivo ejecutable: código, datos, cabecera, información del depurador y tabla de activación imagen.

Además, cada proceso posee su propia tabla de implantación de páginas (TIP), a dos niveles. El objetivo de esa tabla (también llamada tabla de traducción) es, dada una dirección virtual, devolver su dirección física asociada. A veces la dirección virtual no tiene correspondencia en memoria física. Entonces se dice que se ha producido un fallo o defecto de página (page fault). En el siguiente apartado vamos a describir cómo funciona el VMM y qué hace cuando se dan los fallos de página.

B. Funcionamiento del VMM

Cada proceso tiene asignado un número que indica el máximo de páginas físicas que se le pueden conceder. Dicho número es ajustado por el llamado gestor del conjunto de trabajo, del que luego hablaremos. Además, cada proceso lleva asociada una lista que contiene referencias a las páginas físicas a las que se ha accedido menos últimamente.

Cuando el proceso accede a una dirección que no tiene dirección física asociada en la TIP, se produce un fallo de página. Entonces, el VMM consulta el número de páginas que el proceso tiene asignadas. Si no ha llegado al límite, se le concede una nueva página física y se escribe con la correspondiente desde el disco. Esto se denomina paginación bajo demanda. Si por el contrario ya había llegado al límite, entonces hay que descargar una página física al disco para subir a memoria la que ocasionó el fallo. El VMM elige la página víctima de la lista de menos recientemente usadas del proceso.

Nótese que aunque existan páginas libres en la memoria, si el proceso agota su número de páginas asignadas, se producirá el swapping, y además de una de sus propias páginas. Este mecanismo puede parecer ilógico, pero presenta tanto ventajas como inconvenientes. Como problema puede nombrar por ejemplo, que si un proceso estuviera gran parte del tiempo dormido (en espera de un suceso o de una E/S), dicho proceso estaría ocupando páginas físicas que de otro modo podrían ser descargadas. Como ventaja tenemos que, con este esquema, procesos que requieran muchos recursos no dejarán fuera de juego a aquellos cuya demanda de memoria sea escasa. No obstante, el problema planteado es paliado (al menos en gran medida) por una parte del VMM llamada gestor del conjunto de trabajo. Realiza dos funciones fundamentales:

- Por un lado, periódicamente revisa las estadísticas sobre el uso de la CPU por cada proceso. Siguiendo este criterio, procede a ajustar el número que indica el máximo de páginas físicas asociadas a cada uno. A aquellos procesos con poca actividad se les bajará el número (lo que acarreará un descargue de sus páginas físicas, si las tiene, cuando sean necesarias), y a aquellos con mucha carga se les subirá el número.

2. Por otro lado, y también de forma periódica, roba a los procesos páginas físicas, elegidas de entre las menos recientemente usadas por cada uno. Este procedimiento se reliza a la frecuencia necesaria para que los procesos no se ralenticen demasiado por los fallos de página. El objetivo de esto es mantener una reserva de páginas para que una repentina demanda no ocasione una caída en prestaciones del sistema completo. Por ejemplo, cuando se inicia un proceso nuevo se suele requerir una cantidad considerable de páginas.

Para llevar a cabo estas tareas, las páginas físicas se clasifican en una de cuatro listas que son mantenidos por el gestor del conjunto de trabajo:

- lista de páginas modificadas
- lista de páginas no activas
- lista de páginas liberadas
- lista de páginas a cero

Cada página que el gestor roba a un proceso es borrada su lista de menos recientemente usadas e incluida en la lista de modificadas (y las entradas correspondientes de la TIP marcadas como no válidas, de forma que se produzca fallo de página al acceder el proceso a dichas direcciones). Dicha lista contiene páginas robadas que aún están en memoria RAM pero que no han sido escritas a disco (al archivo de paginación). Cuando se produzca un fallo de página, si la dirección física correspondiente estuviera en una de estas páginas, simplemente se volvería a insertar en la lista del proceso y se ajustaría su TIP.

Cuando la lista de modificadas se hace suficientemente grande, otra parte del VMM (el escritor de páginas) copia algunas páginas de la lista al archivo de paginación, las borra de la lista de modificadas y las añade a la lista de inactivas. Ahí están las páginas que han sido robadas, que están en RAM y además en el archivo de paginación. El tratamiento de un fallo de página aquí también sería muy rápido y simple.

Cuando un proceso libera memoria, sus páginas físicas asociadas se añaden a la lista de liberadas. Son, por tanto, potencialmente utilizables sin necesidad de escribirlas en el fichero de swapping pero su contenido no ha sido borrado (están tal y como las dejó el proceso propietario). Periódicamente, estas páginas van siendo inicializadas con ceros,y añadidas a la lista de páginas a cero. El mecanismo de inicialización es para proteger la intimidad del antiguo proceso propietario. Cualquier página que se entrega a un proceso ha de haber sido convenientemente inicializada.

Cuando un proceso requiere memoria física, el VMM comienza cogiendo páginas de la lista de inicializadas. Cuando está vacía, toma de la lista de liberadas, y las inicializa. Cuando ésta se vacía, toma de la lista de inactivas, y las inicializa. Sólo como última opción recurre a las modificadas. Estas últimas requieren escritura en el archivo de paginación junto con inicialización, lo cual es un proceso lento.

En ambientes en los que la memoria es escasa, el gestor del conjunto de trabajo se centra en mantener un conjunto aceptable de páginas disponibles, más que en revisar las estadísticas de uso de la CPU.

Como valores aproximativos, podemos señalar que si el número de páginas a cero más las liberadas más las inactivas suman menos de 20, el gestor robará páginas a procesos que comparten la CPU. Si el número de modificadas supera las 30, procederá a descargar algunas a disco, pasándolas a inactivas.

C. Archivos Asignados en Memoria

Estudiemos ahora cómo el SO utiliza esta facilidad para cargar el código de un ejecutable y sus bibliotecas DLL asociadas.

Un archivo asignado en memoria es todo aquel archivo para el que se ha reservado una región del espacio de direcciones virtuales de un proceso. Puede estar asignado el archivo completo o sólo una porción del mismo (llamada vista).

En principio, el VMM no asigna ninguna página física para el archivo asignado en memoria. El proceso simplemente supone que en ciertas direcciones de su espacio tiene cargado el fichero, así que cuando acceda a alguna se producirá un fallo de página. Es entonces cuando el VMM le asigna algunas páginas físicas y las copia desde el disco (paginación bajo demanda, como comentábamos antes).

La gestión del VMM de los archivos asignados en memoria es como cualquier otra región del espacio direccionable del proceso, excepto que el swapping se hace directamente sobre el archivo (o sea, el archivo de intercambio es el propio archivo asignado en memoria).

Cuando se arranca un proceso con su código grabado en un fichero, el VMM asigna automáticamente dicho fichero en el espacio de direcciones del proceso. También asigna todas las bibliotecas incluidas explícitamente y todas aquellas a las que se hace referencia en el código. Dentro del ejecutable existe una tabla llamada tabla de activación imagen, incluida por el enlazador, cuyas entradas contienen las funciones de biblioteca que se llaman durante el código. Una vez que se cargan las DLL (bibliotecas) en el espacio, el VMM completa la tabla escribiendo para cada entrada la dirección que ocupa la correspondiente función en el espacio del proceso. Por tanto, cada llamada a función implica una búsqueda en la tabla.

Supongamos ahora que se arranca una segunda instancia del mismo proceso. Entonces el VMM pagina en el espacio de direcciones del nuevo proceso el fichero y las DLL, pero no vuelve a asignar páginas físicas, sino que ambas instancias comparten todo, al menos en principio. La ventaja de esto es ahorrar memoria, pero el inconveniente más claro es que si uno de los procesos modificara, por ejemplo, alguna variable global de su segmento de datos, el otro la tendría igualmente modificada. Esto ocurre evidentemente también él la pila e incluso en el mismo código (por ejemplo al ejecutar un depurador sobre una de las instancias para meter puntos de ruptura, se modificaría el código, lo cual implicaría que en la otra también).

Para solucionar el problema, Windows NT (y también UNIX) tiene una propiedad denominada "copiar antes de escribir". El VMM intercepta cualquier instrucción de escritura en el archivo mapeado en memoria por parte de las instancias. Cuando ocurre, asigna una o varias páginas físicas para la instancia escritora, y copia los contenidos de las páginas originales en las nuevas. A partir de ahora, esa instancia posee su propia región del archivo para modificar a su antojo. Análogamente para datos y pila.

UNIX también trabaja así, aunque duplica las zonas de datos y pila por defecto, de forma que múltiples instancias comparten, en principio, sólo el código.

Con respecto a la tabla de activación imagen, decir que el VMM asigna las DLL por defecto en direcciones fijas dentro del espacio de direcciones del proceso. Así, múltiples instancias del mismo pueden compartir la misma tabla. No obstante, un proceso puede especificar la dirección base de cada DLL; en ese caso, el proceso tendría su propia tabla en memoria, con lo que resulta más eficiente que todos los procesos tengan las DLL asignadas en las mismas direcciones.

Desde el punto de vista del programador, se pueden asignar en memoria archivos de hasta 264 bytes usando vistas del archivo.

El archivo se abre con CreateFile. Los primero es crear un objeto de asignación de archivo para él, con la función CreateFileMapping indicando el tamaño real del archivo. Después podemos definir diferentes vistas,

con CreateViewOfFile. La vista se abandona con UnmapViewOfFile (y fueza al VMM a escribir las modificaciones en disco).

Los archivos asignados en memoria son el único mecanismo de compartición de datos entre procesos. Existe un archivo que crea el objeto de asignación, y el resto de los procesos usarán el mismo objeto referenciado por la función OpenFileMapping. Si como descriptor de archivo se pasa a las funciones 0xFFFFFFFF, entonces el VMM usará el archivo de paginación como medio de compartición de datos.

En una red no es posible usar este mecanismo de compartición de datos, pues el SO no garantiza la coherencia de los mismos. Por ejemplo, una CPU en un ordenador podría modificar el archivo en disco y otra en otro distinto tenerlo en memoria, con lo cual no se percataría de la actualización.

D. Uso de Memoria Virtual por parte del Programador

Vamos a ofrecer unas pinceladas sobre cómo el programador puede hacer uso explícito del mecanismo de la memoria virtual, y así hacer aplicaciones más eficientes. Se trata sin duda de una forma elegante de programar, ya que permite situar objetos de grandes dimensiones en el espacio de direcciones del proceso, cuyo tamaño real se desconoce al tiempo de compilación. Por ejemplo, una hoja de cálculo.

Consiste en dos etapas: la reserva y la asignación.

La reserva no es más que indicar al SO que una región del espacio de direcciones del proceso se va a destinar a un determinado objeto u objetos. Por tanto no hay ninguna correspondencia con memoria física. El VMM toma nota en un VAD (Virtual Address Descriptor) de la dirección de inicio, número de bytes reservados, y protección de la zona (en efecto, las páginas pueden tener permisos de lectura, lectura/escritura, o ninguno). Si el proceso accede a una de estas direcciones, se elevará una excepción. La reserva se hace siempre por trozos de 64 Kb.

La asignación consiste en hacer corresponder toda o parte de la memoria antes reservada con memoria física. No obstante, el VMM no va a asignar memoria física inmediatamente, sino que simplemente actualizará el VAD para permitir que esa memoria sea ahora accesible, y se asegurará que en el archivo de paginación haya espacio suficiente. Cuando el proceso haga un acceso, se producirá un fallo de página, y es en ese momento cuando el VMM asignará páginas físicas (para la página que produzca el fallo y las vecinas) y actualizará la TIP del proceso.

El problema de usar memoria virtual explícitamente es que el programador debe llevar una lista de la memoria que ha asignado de entre la que ha reservado. Otra solución es instalar un manejador de excepciones, de manera que al acceder a una dirección reservada pero no asignada se eleve una excepción. El manejador de esa excepción hará la asignación y se continuará la ejecución.

Para reservar memoria se llama a la función VirtualAlloc pasándole el indicador MEM_RESERVE, y para asignar, pasándole MEM_COMMIT.

La memoria reservada o asignada se puede liberar con VirtualFree, pero entonces ha de liberarse por completo.

Con VirtualLock indicamos al VMM que no descargue una determinada página a disco cuando el proceso esté ejecutándose (aunque cuando no se ejecute perdamos el control).

Por último señalar que existen funciones del API Win32 que permiten verificar si una dirección virtual tiene correspondencia física o no.

E. Bloques (heaps)

Los bloques de memoria son muy útiles cuando el programador no necesita usar memoria virtual explícitamente. Podemos tener estructuras de datos distintas en bloques distintos de memoria, de manera que un fallo en la manipulación de estructuras de un tipo no repercuta en las demás.

Además, de este modo se gestionaría más eficientemente la memoria, ya que al borrar estructuras y escribir otras no se provocaría fragmentación interna dentro del bloque (al ser todas las de un tipo del mismo tamaño). Una última razón sería él poder minimizar los fallos de página. Estructuras en un mismo bloque son probables que compartan la misma página, luego podríamos acceder a todas las de un mismo bloque con sólo un fallo de página.

V. SISTEMA DE ARCHIVOS

Windows NT soporta cuatro tipos de sistemas de ficheros (SF) disintos, y puede trabajar con los cuatro a la vez (un disco con varias particiones, en cada una un SF distinto). Son los siguientes:

Sistema de Archivos	Sistemas Operativos Soportados
FAT	DOS, Windows NT, y OS/2
HPFS	OS/2 y Windows NT
NTFS	Windows NT
CDFS	Windows NT

NTFS (New−Technology File System): Es el sistema de ficheros nativo de Windows−NT. Como características podemos señalar:

- Permite nombres de archivo de hasta 255 caracteres, sensibles al tipo de letra.
- Permite la gestión de medios de almacenamiento extraordinariamente grandes.
- Incorpora mecanismos para garantizar la seguridad.
- Soporta el concepto de enlace (por compatibilidad con el estándar POSIX).
- Es capaz de recomponerse rápidamente después de una caída del sistema.
- Soporta el estándar Unicode.

FAT (File Allocation Table): El que usa MS−DOS y Windows 16 bits. Es el SF más pobre, y es se mantiene para dar soporte a las aplicaciones DOS.

Figura 3. Propiedades de los Sistemas de Archivos FaT y NTFS

HPFS (High−Performance File System): Es el que usa el sistema operativo OS/2. Se ha incluido para dar soporte a las aplicaciones OS/2 y complementar así al subsistema del mismo nombre. No es capaz de recomponerse del todo bien después de una caída del sistema ni de asegurar la no corrupción de los datos.

CDFS (CD−ROM File System): Es un SF que Microsoft ha desarrollado exclusivamente para montarse sobre los CD−ROM.

Vamos a centrarnos en el más importante de los cuatro: NTFS. Este sistema de ficheros lleva incorporados muchos conceptos de teoría de bases de datos relacionales.

Proporciona la seguridad, recuperación y tolerancia a fallos en base a la redundancia de datos. De hecho, implementa los cinco primeros niveles del pseudo−estándar RAID. RAID significa Redundant Arrays of Inexpensive Disks, algo así como "vectores redundantes de discos baratos". Actualmente, el coste de los

almacenamientos masivos o secundarios es ínfimo, y ya se pueden encontrar discos de varios Gb por menos de S/.500.00.

A eso se refieren, a mi entender, las siglas. RAID es una "norma" de facto que se basa fundamentalmente en conseguir la integridad de los datos a base de dividirlos en pedazos y repartir los pedazos entre varios discos, junto con informaciones redundantes de comprobación de errores. Cada nivel ofrece una estrategia distinta para conseguir la integridad. Los niveles son:

- **Nivel 0:** los datos de cada fichero se dividen en porciones, las cuales se reparten en un orden fijo entre los distintos discos con los que cuente el sistema. Realmente aquí no se usan códigos de comprobación de errores.

- **Nivel 1:** de cada disco se realiza una copia o espejo. Es la estrategia que da mayor fiabilidad (pero también, evidentemente, la más costosa).

- **Nivel 2:** como el 0, pero un algortimo va construyendo una serie de códigos correctores de errores. Esos códigos son igualmente distribuidos entre unos discos destinados especialmente a ese uso.

- **Nivel 3:** como el 2, pero no se usan códigos correctores sino un simple código de paridad, que puede ser guardado en un mismo disco para todos los ficheros.

- **Nivel 4:** como el 3, pero dividiendo los ficheros en segmentos de datos más grandes.

- **Nivel 5:** es el nivel más usual; es igual que el4, pero no usa un disco separado para almacenar los códigos de paridad, sino que divide igualmente esos códigos y los distribuye por los disco, intentando que no coincidan en la misma zona de disco datos de un fichero con sus códigos de paridad correspondientes.

- **Nivel 6:** igual que el 5 pero se auxilia de elementos hardware, tales como controladoras de disco especiales, fuentes de alimentación.

En NTFS, al igual que en los SF de UNIX, existe una serie de permisos sobre ficheros y directorios, que son los siguientes: lectura (R), escritura (W), ejecución (X), borrado (D), cambio de permisos (P) y ser el nuevo propietario (O). Todo fichero y directorio tienen un propietario, que puede conceder permisos sobre ellos. El Administrador del sistema puede tomar la propiedad de cualquier fichero o directorio sobre NTFS, pero no transferirla de nuevo a ningún otro usuario, a diferencia de UNIX, ni siquiera a su dueño original.

Sistemas de Archivos	Ventajas	Desventajas
FAT	Poco consumo de sistema. El mejor para discos y/o particiones de menos de 200MB.	El rendimiento decrece con particiones de más 200MB. No se pueden aplicar permisos sobre archivos y directorios.
HPFS	El mejor para discos y/o particiones entre 200 y 400 MB. Elimina la fragmentación almacenando en un solo bloque el archivo completo.	No es eficiente para menos de 200MB.No soporta hot fixing. No se pueden aplicar permisos sobre archivos y directorios.
NTFS	El mejor para volúmenes de 400MB o más. Recuperable (registro de transacciones), diseñado para no ejecutarle utilerias de reparación. Es posible establecer permisos y registro de auditoría sobre archivos y	No recomendable para volúmenes de menos de 400MB. Consume de 1 a 5 MB de acuerdo al tamaño de la partición.

	directorios.	

Ventajas y Desventajas de los Sistemas de Archivos

A continuación vamos a comentar las llamadas al sistema más usuales para crear ficheros, leer de ellos, escribir en ellos, etc.

A. Creación/Apertura de Ficheros

Para crear/abrir un fichero se usa la llamada al sistema

HANDLE CreateFile (LPTSTR lpszName, DWORD fdwAccess, DWORD fdwShareNode, LPSECURITY_ATTRIBUTES lpsa, DWORD fdwCreate, DWORD fdwAttrsAndFlags, HANDLE hTemplateFile);

lpszName: nombre del archivo a crear o abrir.

fdwAccess: especifica el modo de acceso al archivo: lectura (GENERIC_READ), escritura (GENERIC_WRITE), o ambos.

fdwShareMode: permisos que tendrá la compartición del archivo a abrir: 0 (ningún proceso podrá abrirlo hasta que nosotros lo cerremos), FILE_SHARE_READ (otros procesos pueden abrirlo pero sólo para leer), FILE_SHARE_WRITE (sólo para escribir; no se suele usar), o una combinación de ambos.

lpsa: la típica estructura de seguridad de todo objeto en Windows NT. Sólo tendrá sentido si el archivo se crea en un SF que soporte seguridad, como NTFS.

fdwCreate: una serie de indicadores que especifican una acción:

CREATE_NEW: crea un archivo nuevo, y da error si ya existe

CREATE_ALWAYS: crea un archio nuevo,y si existe lo machaca

OPEN_EXISTING: abre un archivo, y da error si no existe

OPEN_ALWAYS: abre un archivo y si no existe lo crea

TRUNCATE_EXISTING: si el archivo exisye, lo abre pero truncando su tamaño a 0 bytes; si no existe, da una error.

fdwAttrsAndFlags: sirve para dar atributos al fichero (sólo si lo estamos creando) y activar ciertas banderas. Veamos algunos.

Atributos:

FILE_ATTRIBUTE_HIDDEN: archivo oculto

FILE_ATTRIBUTE_NORMAL: archivo sin atributos especiales

FILE_ATTRIBUTE_READONLY: archivo de sólo lectura

FILE_ATTRIBUTE_SYSTEM: archivo del sistema

FILE_ATTRIBUTE_TEMPORARY: archivo temporal; el Executive intentará mantenerlo en RAM tanto como le sea posible

FILE_ATTRIBUTE_ATOMIC_WRITE: para indicar que los datos de este archivo son críticos; eso hará que el Executive aumente la frecuencia con que escribe estos datos de RAM a disco.

A propósito de los dos últimos indicadores, comentar que el Executive no escribe a disco inmediatamente los cambios realizados a un archivo en RAM, pues degradaría las prestaciones del sistema. Usa un mecanismo de escritura diferida, de manera que se escribe a disco cuando se cierra el archivo, o cuando el sistema está desocupado, o cuando es necesario hacer swapping. Para los ficheros cuyos datos son críticos, necesitamos que las modificaciones sean escritas rápidamente a un soporte no volátil, por si el sistema se cayera.

Banderas:

FILE_FLAG_NO_BUFFERING: con esta bandera le indicamos al Executive que no gestione buffers de memoria con relación a la entrada/salida de este archivo, sino que lea y escriba directamente a disco.

FILE_FLAG_RANDOM_ACCESS: queremos acceso directo al archivo.

FILE_FLAG_SEQUENTIAL_SCAN: acceso secuencial.

FILE_FLAG_WRITE_THROUGH: con este indicador, el SO enviará a disco los datos que hayan sido modificados en memoria, pero los mantendrá en memoria para acelerar las lecturas

FILE_FLAG_POSIX_SEMANTICS: acceso al archivo según el estándar POSIX (por ejemplo, sensible al tipo de letra)

FILE_FLAG_BACKUP_SEMANTICS: cuando un proceso solicita abrir un archivo, el SO normalmente realiza ciertos tests de seguridad para comprobar si el proceso tiene o no los permisos necesarios. Con este indicador se anulan ciertos tests, de manera que el SO comprueba si el proceso tiene permiso para acceder al archivo, y si es así, le permite el acceso pero sólo para realizar una copia de seguridad. Aunque tenga permiso de escritura, le será anulado.

FILE_FLAG_OVERLAPPED: los accesos a los archivos suelen hacerse de manera síncrona (el subproceso duerme hasta que se consuma el acceso). Con este indicador se permite realizar E/S asíncrona, con lo que el subproceso seguirá ejecutándose y el SO le informará cuando la E/S finalice.

hTemplate: el descriptor de un archivo ya abierto; si no es NULL, los atributos y banderas de dicho archivo serán asignados a los de nuestro archivo.

La llamada retorna el descriptor al objeto archivo, o −1 si hubo algún error.

B. Cierre de Ficheros

Se usa la misma llamada que para cerrar un descriptor a cualquier objeto. El Sistema Operativo decrementará el contador de uso del archivo, y si es 0 lo cerrará definitivamente.

BOOL CloseHandle (HANDLE hObject);

C. Lectura/escritura a ficheros

Windows NT permite que los subprocesos hagan E/S a ficheros de manera síncrona o asíncrona. EL modo

síncrono es el habitual: un subproceso inicia una operación de E/S sobre un fichero; el Executive lo pondrá a dormir hasta que esa E/S de complete. En cambio, en el modo asíncrono, el subproceso que inicia la operación puede seguir ejecutándose, y cuando necesite los datos de la E/S se pondrá voluntariamente a esperar, de manera que si para ese tiempo la E/S se ha completado, obtendrá los resultados inmediatamente.

Se usan las mismas llamadas al sistema para ambos tipos de acceso. En dichas llamadas existirá un parámetro de entrada (lpOverlapped) que, si es NULL, indicará que la llamada es acceso síncrono; si no, indicará acceso asíncrono, dando la dirección de una estructura OVERLAPPED que tiene el siguiente formato:

typedef struct _OVERLAPPED{

DWORD Internal;

DWORD InternalHigh;

DWORD Offset;

DWORD OffsetHigh;

DWORD hEvent;

} OVERLAPPED;

Internal: cuando la E/S se completa, el sistema guarda en esa palabra un estado interno.

InternalHigh: cuando la E/S se completa, el sistema guarda ahí el número de bytes transferidos.

Offset,OffsetHigh: indican la posición del byte del archivo donde queremos comenzar el acceso.

hEvent: el descriptor (opcional) de un objeto suceso.

Vamos a suponer que un subproceso A desea realizar E/S asíncrona sobre un fichero, y para ello inicializa el parámetro lpOverlapped de la llamada con la dirección correspondiente a una estructura OVERLAPPED. Entonces sigue ejecutándose. Cuando al Executive le llega la petición de E/S sobre el fichero, pone el estado de este objeto a no señalado. Cuando la E/S finaliza, lo pone a señalado (esto ya lo vimos en el capítulo de los procesos). En algún momento, A necesita los datos de la E/S que inició, con lo que se pone a esperar a que el objeto archivo se ponga a estado señalado (para lo cual usará una llamada tipo WaitFor...Object(s), como ya explicamos). Si, en el momento de hacer la llamada a E/S, el subproceso A hubiera especificado en hEvent un descriptor a un suceso, el Executive pondría a señalados tanto el objeto fichero como dicho objeto suceso, con lo cual el subproceso A tendría la facilidad de esperar por cualquiera de los dos.

La utilidad de esto es en la situación de que el fichero es compartido, y varios subprocesos pueden estar haciendo E/S sobre él y, por consiguiente, poniéndose a estado señalado/no señalado múltiples veces. Especificando un objeto suceso propiedad de A, dicho subproceso sabrá que cuando el suceso esté señalado, seguro que se ha completado su operación de E/S y no la de otro.

Una vez explicados estos matices, pasamos sin más a describir el perfil de las llamadas de lectura/escritura, que son muy parecidas a las que usa UNIX:

BOOL ReadFile (HANDLE hFile, LPVOID lpBuffer, DWORD nNumberOfBytesToRead, LPWORD lpNumberOfBytesRead, LPOVERLAPPED lpOverlapped);

hFile: es el descriptor al archivo.

lpBuffer: apunta a un área de memoria donde la llamada colocará los datos leídos

nNumberOfBytesToRead: es el número de bytes a leer.

nNumberOfBytesRead: es un parámetro de salida que indica el número de bytes que se leyeron en realidad.

lpOverlapped: elige modo síncrono/asíncrono; ya comentado

BOOL WriteFile(HANDLE hFile, CONST VOID * lpBuffer, DWORD nNumberOfBytesToWrite, LPDWORD lpNumberOfBytesWritten, LPOVERLAPPED lpOverlapped);

hFile: es el descriptor al archivo.

lpBuffer: apunta a un área de memoria donde se encuentran los datos a escribir.

nNumberOfBytesToWrite: número de bytes a escribir.

nNumberOfBytesWritten: número de bytes escritos en realidad.

lpOverlapped: elige modo síncrono/asíncrono; ya comentado.

Nos gustaría señalar que en los accesos síncronos existe un puntero de lectura/escritura sobre un archivo para cada subproceso que esté accediendo al mismo. Ese puntero lo asigna el SO en el momento de apertura del archivo cuando un subproceso lo abre para accesos síncronos (o sea, sin especificar el indicador FILE_FLAG_OVERLAPED). El puntero se modifica al leer y al escribir. En el caso de que el acceso síncrono sea también acceso directo, entonces es posible cambiar el puntero usando la llamada SetFilePointer, que no merece la pena comentar.

Nótese que en los accesos asíncronos no existe el puntero, por lo que hemos de indicar la dirección de inicio de cada acceso (recordemos que se indica en la estructura OVERLAPPED).

Existen unas llamadas de lectura/escritura extendidas (ReadFileEx y WriteFileEx), que son muy útiles a la hora de realizar una E/S asíncrona. Permiten que se les pase la dirección de una función de forma que, cuando la E/S asíncrona se complete, se salte a la ejecución de esa función. Para ello, el subproceso ha de estar esperando por el fin de la E/S con una función WaitFor...Object(s) extendida (WaitForSingleObjectEx o WaitForSingleObjectsEx). De hecho, WaitForSingleObject está construida internamente como una llamada a WaitForSingleObjectEx pasándole un parámetro que indica que no queremos uso extendido.

D. Atributos de Ficheros

Los atributos que se indican al crear un fichero en el parámetro fwdAttrsAndFlags pueden ser consultados con una llamada a:

BOOL GetFileInformationByHandle (HANDLE hFile, LPBY_HANDLE_FILE_INFORMATION lpFileInformation);

Esta llamada devuelve en la estructura apuntada por lpFileInformation toda la información relativa al fichero cuyo descriptor es hFile. La estructura tiene los siguientes campos:

typedef struct _BY_HANDLE_FILE_INFORMATION {

DWORD dwFileAtributes;

FILETIME ftCreationTime;

FILETIME ftLastAccessTime;

FILETIME ftLastWriteTime;

DWORD dwVolumeSerialNumber;

DWORD nFileSizeHigh;

DWORD nFileSizeLow;

DWORD nNumberOfLinks;

DWORD nFileIndexHigh;

DOWRD nFileIndexLow;

} BY_HANDLE_FILE_INFORMATION;

dwFileAttributes: los atributos del fichero que se pasaron en el parámetro fdwAttrsAndFlags en el momento de su creación (ver CreateFile)

ftCreationTime: fecha de creación del fichero

ftLastAccessTime: fecha del último acceso al fichero

ftLastWriteTime: fecha de la última escritura al fichero

dwVolumeSerialNumber: número de serie del volumen donde se encuentra el fichero

nFileSizeHigh, **nFileSizeLow**: 64 bits que indican el tamaño del fichero; por tanto, Windows NT permite ficheros de hasta 264 bytes

nNumberOFLinks: número de enlaces del fichero; este parámetro asegura la compatibilidad con el estándar POSIX. Recordemos que en UNIX cada fichero tiene un número de enlaces que indican los distintos nombres que referencian al mismo fichero dentro del sistema de ficheros. Así se evita la redundancia de datos.

nFileIndexHigh, nFileIndexLow: es un identificador único que el Executive asocia a un fichero en el momento de que algún subproceso lo abre. Si el sistema se apaga y se enciende, el identificador puede no ser el mismo. Sin embargo, en la misma sesión, procesos distintos leerán el mismo identificador. Esto es útil para determinar si dos descriptores distintos referencian en realidad al mismo fichero dentro de un volumen (basta hacer sendas llamadas a GetFileInformationByHandle y ver si el identificador y número de volumen coinciden en las estructuras devueltas por cada una).

E. Bloqueo de Archivos

Otra llamada muy importante es la que permite el bloqueo de todo o parte de un fichero, de manera que ningún otro subproceso pueda acceder a la región bloqueada.

La llamada para bloquear es:

BOOL LockFile (HANDLE hFile, DWORD dwFileOffsetLow, DWORD dwFIleOffsetHigh, DWORD cbLockLow, DWORD cbLockHigh);

hFile: descriptor al archivo a bloquear.

dwFileOffsetLow, **dwFileOffsetHigh**: dirección de comienzo de la región a bloquear.

cbLockLow, cbLockHigh: tamaño de la región a bloquear.

La llamada para desbloquear es:

BOOL UnlockFile (HANDLE hFile, DWORD dwFileOffsetLow, DWORD dwFIleOffsetHigh, DWORD cbUnlockLow, DWORD cbUnlockHigh);

Los parámetros son análogos a los anteriores. Es necesario que un subproceso desbloquee un área que previamente bloqueó antes de que finalice. En caso contrario, estará impidiendo el acceso al fichero a todos los demás subprocesos.

Existen versiones extendidas de ambas llamadas (LockFileEx y UnlockFileEx) que permiten establaces bloqueos pero sólo de escritura (otros subprocesos podrán leer el área bloqueada, pero no escribir en ella).

VI. ENTRADA Y SALIDA

A lo largo de los capítulos precedentes hemos hablado de los distintos aspectos de la E/S: gestión de las caches, E/S síncrona y asíncrona, drivers de dispositivo, nivel de abstracción de hardware, ficheros y sistemas de ficheros, por citar algunos. Cuando describíamos el administrador de E/S del Executive, hablamos también de que se encargaba de gestionar los mecanismos relacionados con la red. Me parece conveniente dedicar parte de este último capítulo al tema de las comunicaciones en Windows NT, por tratarse de uno de los aspectos donde más fuertemente brilla este SO.

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

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

- **En el nivel 0** aparece un dispositivo que es la tarjeta de interfaz a la red (Network Interface Card, NIC). El NIC conecta el bus interno de la máquina con la red, sirviendo de interfaz entre el nivel 0 y el 1 (nivel físico). Es contemplado por el SO como un periférico más, controlado a través de su driver correspondiente.

- **En el nivel 2** (nivel de enlace de datos) aparece un software llamado NDIS (Network Device Interface Specification), que es una interfaz entre los drivers de dispositivo del NIC y los protocolos de transporte.

- **En los niveles 3** (nivel de red) **y 4** (nivel de transporte) Windows NT sitúa el software de los protocolos de transporte. Soporta TCP/IP, NBF, NWLink, DLC y AppleTalk.

- **En el nivel 5** (de sesión) aparecen dos interfaces con los protocolos de transporte, que son los

Windows Sockets (WinSock) y la NetBIOS.

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

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

En la misma capa de sesión están dos subsistemas integrales gestionados por el administrador de la E/S, denominados el redireccionador y el servidor. El redireccionador es el responsable de enviar peticiones de E/S a lo largo de la red, cuando el fichero o dispositivo solicitados son remotos. Al servidor le llegan peticiones desde los redireccionadores clientes, y las gestiona de modo que alcancen su destino. Tanto servidores como redireccionadores son implementados como drivers del sistema de ficheros. Así, cuando un proceso quiere realizar una E/S, usará las mismas llamadas al sistema para acceso local o remoto, con lo que no necesita conocer la ubicación del recurso (fichero o dispositivo). Pueden existir múltiples parejas de redireccionadores−servidores ejecutándose concurrentemente.

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

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

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

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

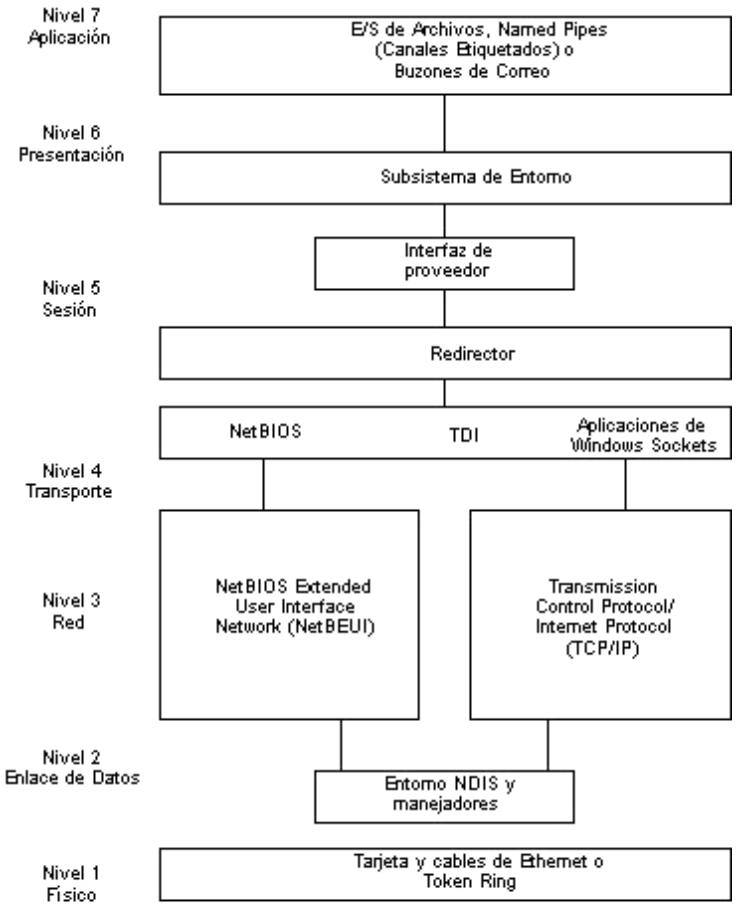

Figura 4. Windows NT y el Modelo OSI

VII. WINDOWS NT 5

En la versión 4.0, Microsoft nos dio una alegría al cambiar la interfaz gráfica de Windows NT y sustituirla por Indy, la bonita GUI de Windows 95. Desgraciadamente, NT 4.0 no incluía muchas de las cosas que se venían anunciando desde hacía tiempo, lo que nos dejó un sabor agridulce.

En Windows NT 5.0, Microsoft da un paso de gigante, incluyendo las siguientes novedades importantes:

- Servicios de Directorio al estilo de X.500.
- Modelo de Objetos de Componentes Distribuidos (DCOM).
- Sistema de seguridad Kerberos.

A. Servicios de Directorio: Active Directory

NT 5.0 incluye un servicio de directorio llamado Active Directory, basado en DNS (Domain Name Server) y LDAP (Lightweight Directoy Access Protocol).

El protocolo DNS da una manera de nombrar máquinas situadas en cualquier parte del planeta y nos permite conocer sus correspondientes direcciones IP, gracias a que estructura los nombres de las máquinas jerárquicamente por dominios, y cada dominio conoce potencialmente las direcciones IP de las máquinas que pertenecen a él. Si quiero conocer la dirección de la máquina mi_servidor.dom1.edu, preguntaré a algún

servidor del dominio edu, el cual, o la sabe directamente, o preguntará a algún servidor de dom1, y así.

Con Active Directory vamos a poder localizar cualquier objeto llamándolo por su nombre, y acceder a información sobre él. Un objeto será algo heterogéneo: una máquina en Internet, un fichero, o un proceso en ejecución. La información sobre ese objeto dependerá de la clase a la que pertenezca dicho objeto (por tanto será también algo heterogéneo). El pegamento que aglutina todo esto se llama Active Directory.

La idea no es nueva: el sistema de directorios X−500 permite algo parecido, pero su complejidad ha hecho que no esté muy difundido entre los sistemas operativos. Por ello se desarrolló LDAP, una versión simplificada de X−500.

En Active Directory va a tener, para cada dominio de nuestra LAN, un nombre de dominio al estilo DNS, y uno o varios servidores de dominio (llamados controladores de dominio). Cuando LDAP sea, al igual que DNS, un estándar, podremos acceder a la base de datos de directorios del servidor Active Directory usando un cliente UNIX, OS/2 o Macintosh.

Tener varios controladores de dominio asociados al mismo dominio interesa cuando necesitemos alto rendimiento y baja tasa de errores. Cada controlador va a almacenar la misma base de datos del dominio del directorio. Active Directory asegura la integridad de esas BD, de manera que actualizar una implique la actualización de cada una de sus copias. Para ello se usa un protocolo de comunicación entre controladores. La actualización de las copias se realiza sólo sobre los datos modificados.

La replicación se realiza vía RPCs cuando estamos en un ámbito local, con alta fiabilidad y baja tasa de errores. Para redes a través de líneas telefónicas, se ha incluido la opción del correo electrónico como método para que los controladores se intercambien información de replicación.

En una LAN, la replicación se produce cada 5 minutos. En una WAN, el administrador puede ampliar el intervalo para aumentar así las prestaciones.

Existen varias alternativas a la hora de elegir una API para los clientes Active Directory, destacando ADSI (Active Directory Sservice Interface).

B. Modelo de Objetos de Componentes Distribuidos (DCOM)

DCOM es una extensión natural a COM. COM es un estándar para la comunicación entre objetos independientemente del lenguaje en el que hayan sido escritos (por ejemplo, objetos Java con obejtos C++). El objeto cliente accederá a los métodos del objeto servidor a través de interfaces COM normalizados. Quizás nos suene más el nombre de componentes ActiveX.

DCOM extiende lo anterior a un ámbito de red; los objetos a comunicarse no tienen porqué compartir la misma máquina.

Pero esto no es nuevo: lo podíamos hacer desde hacía tiempo con las RPC. De hecho, DCOM está construído sobre las RPC; se trata de un estándar de más alto nivel, con el que podremos escribir aplicaciones distribuidas en un entorno de red sin necesidad de conocer todos los entresijos de las RPC, y además con un enfoque orientado a objetos. De hecho, Microsoft también se refiere a DCOM como Object RPC (ORPC).

DCOM se incluyó a Windows NT en su versión 4.0 (también salió en 1.996 una versión para Windows 95), y actualmente la empresa Sofware AG prevé lanzar versiones para Solaris, Linux y otros sistemas a finales de este año. Con esto tenemos que DCOM se está convirtiendo en un estándar importante en la industria, y se podrá utilizar para comunicación entre objetos corriendo en sistemas operativos distintos.

Entonces, ¿qué aporta NT 5.0?

Supongamos que tengo un objeto servidor subsumido en un proceso en mi máquina servidora, y que un cliente quiere acceder a alguno de los métodos que mi objeto exporta como públicos. Entonces, el componente DCOM localiza él solito al objeto servidor en la red, y le manda una RPC. DCOM encuentra el objeto servidor en la red de dos posibles maneras:

En Windows NT 4.0, el cliente ha de conocer dónde está el servidor (lo tiene escrito en el Registro) y se lo dice a DCOM (trivial).

En Windows NT 5.0, DCOM usa Active Directory para hallar la dirección del objeto. Ésta es la gran ventaja sobre el caso anterior. Además de la dirección, puede encontrar más información sobre el objeto servidor.

C. Servicios de Seguridad: el estándar Kerberos

En UNIX, de la seguridad se encarga un módulo llamado Kerberos, desarrollado por el MIT como parte del Proyecto Atenas. Kerberos es actualmente un estándar en la industria, y Microsoft ha implementado la versión 5 de la norma en NT 5.0.

Como sabemos, NT soporta varios protocolos de seguridad. Existe, no obstante, una interfaz común que los aglutina (la SSPI, Security Service Provider Interface), y que proporciona una API común a los niveles superiores. En NT 4.0, la API cubre los protocolos SSL (Secure Sockets Layer, junto con su versión PCT) y NTLM (NT Lan Manager). En NT 5.0, Kerberos se une al grupo. Los usuarios del nivel de seguridad son otros protocolos, que, usando la API SSPI, acceden a los servicios que ofrece ese nivel, eligiendo el protocolo que deseen de entre los de la lista. Ejemplos de protocolos clientes son HTTP, LDAP, CIFS (usado por el Sistema de Ficheros Distribuido) y RPC.

En entornos de red, las aplicaciones usan primordialmente el protocolo NTLM, que da autentificación, integridad de datos y privacidad. Esto va a cambiar con la introducción del estándar Kerberos.

Kerberos, al igual que NTLM, proporciona autentificación, integridad de datos y privacidad. Entre las mejoras que hace a NTLM se encuentra la autentificación mutua, es decir, que tanto el cliente ha de probar su identidad al servidor como el servidor al cliente. Cada dominio de la red va a tener su servidor Kerberos, que utiliza la base de datos de Active Directory, con lo que se habrá de ejecutar sobre la misma máquina que el controlador de dominio. También puede estar replicado dentro del mismo dominio.

De manera muy general, podemos decir que un usuario que desee acceder a servicios de una máquina remota debe primero hacer logon sobre el servidor Kerberos del dominio correspondiente (o sobre alguna de sus copias, si cabe).

Si los procesos de identificación y declaración de privilegios son correctos, Kerberos entrega al cliente un "ticket que concede tickets" llamado TGT (ticket−granting ticket) de acuerdo a los privilegios del cliente. Usando ese ticket, el cliente podrá de nuevo solicitar a Kerberos otros tickets para acceder a determinados servidores del dominio. Kerberos examinará el TGT del cliente y el servicio que desea; si el TGT es válido para acceder al servicio solicitado, Kerberos entregará al cliente un ticket nuevo para acceder al servicio concreto. El cliente envía ese nuevo ticket a la máquina donde se encuentra el servicio al que desea acceder; el servidor posiblemente lo examinará para comprobar la identificación del usuario (autentificación). Los tickets están todos encriptados.

El estándar Kerberos versión 5 no especifica el contenido de los mensajes que se intercambian para identificar al cliente. Por ello, los mensajes de cada implementación de la norma serán distintos, con lo que podríamos tener problemas al interactuar con Kerberos de otros sistemas operativos.

G L O S A R I O

APPC: (Advanced Program to Program Communications). Comunicación Avanzada Programa a Programa.

Arpanet: (Advanced Research Projects Agency NETwork). Red Avanzada de Agencias para Proyectos de Investigación. Red de Investigación fundada por DARPA (Originalmente ARPA) y construida por BBN, Inc., en 1969. Fue pi onera en tecnología de conmutación de paquetes y fue la piedra angular original y la base de la ahora gigantesca Internet. En 1983, la parte militar de comunicaciones se dividió como MILNET.

AT&T: (American Telephone and Telegraph Corporation). Corporación Americana de Telefonía y Telegrafía.

ATM: (Asynchronous Transfer Mode). Modo de Transferencia Asíncrona. Red estándar para transmitir a alta velocidad por medio de fibras ópticas. Utiliza un paquete de 53 bytes de longitud fija para datos.

Bus: Ducto, colector. Es un canal o ruta común entre dispositivos del hardware, ya sea internamente entre componentes de la computadora o externamente entre estaciones de una red de comunicaciones.

Cuando la arquitectura de bus es utilizada en una computadora, el procesador o procesadores, los bancos de memoria y las unidades de control periféricas están todos interconectados mediante el bus. El bus está dividido en dos canales, uno para seleccionar donde está localizado el dato (Bus de direcciones) y otro para transferir el dato (Bus de datos). Cuando se conecta una tarjeta de circuito impreso en una de las ranuras de expansión de una computadora personal, se le está conectando al bus.

Cuando la arquitectura de bus se utiliza en una red, todas las terminales y computadoras están conectadas a un canal común constituido por pares de cables trenzados, cable coaxial o fibras ópticas.

COM: (Computer Output Microfilm). Micropelícula Sacada por Computadora. Máquinas que producen micropelículas o microfichas directamente de la computadora. Las COM pueden ser unidades independientes o en l&iac ute;nea con la computadora y recibir datos de la misma forma que una impresora, ya preparadas con encabezamientos, números de página, etc.

Utilizando métodos que toman una fotografía de la imagen generada en un CRT, o empleando lasers que dibujan directamente la imagen, las unidades COM crean una imagen en película de cada página de un informe. Se pueden agregar asimismo gráficos adicionales, tales como líneas y logos.

Concentrador (Hub): Un dispositivo que une varios canales de comunicaciones en uno solo. Un concentrador es similar a un multiplexor, excepto que no separa las señales en el otro extremo. Es la computadora receptora quien ejecuta esta función.

CPU: (Control Processing Unit). Unidad Central de Proceso (Procesamiento) La parte de una computadora que realiza la computación. También llamada el procesador, está constituida por la unidad de control y la ALU (Unidad Aritmético−Logica).

La CPU de una computadora personal está contenida en un microprocesador único. La CPU de una minicomputadora está contenida en una o varias tarjetas de circuito impreso. La CPU de una macrocomputadora es tá contenida en muchas tarjetas de circuito impreso.

La CPU, el reloj y la memoria principal constituyen una computadora. Un sistema informático completo requiere la agregación de unidades de control, dispositivos de entrada, salida y almacenamiento y de un sistema ope rativo.

Los términos CPU y procesador implican el uso de la memoria principal, como en la frase "se envían datos a la CPU y se los procesa", ya que para poder ser procesados, los datos deben estar almacenados en la memoria.< /P>

Cracker: Pirata informático, intruso informático. Un programador que escribe programas en lenguaje ensamblador o en lenguajes a nivel de sistemas, como C. Si bien esto puede referirse a cualquier programador, implic a un tedioso "desmenuzamiento" de bits y bytes. Algunas veces se refiere a una persona que viola un código y obtiene ingreso ilegal a un sistema.

Chip: Es un circuito integrado. Los chips tienen aproximadamente de 2 a 12 mm. de lado y aproximadamente 1 mm. de espesor. Contienen desde unas pocas decenas hasta varios millones de componentes electrónicos (transistores, resistencias, etc.). Los términos chip, circuito integrado y mircoelectrónica, son sinónimos.

Datagrama: Unidad de mensaje TCP/IP que contiene las direcciones origen y de destino de Internet y los datos.

DCOM.−(Distributed Component Object Model). Modelo de Objeto Común Distribuido.

DHCP: (Dinamic Host Configuration Protocol). Protocolo de Configuración Dinámica de Host.

DNS: (Domain Naming System). Sistema de Nombres de Dominio. Dar nombres de Dominios Sistema de direccionamiento de correo electrónico utilizado en redes como Internet y Bitnet.

EISA: (Extended Industry Standard Architecture). Arquitectura Estándar Industrial Extendida. Estándar de bus para PC que extiende la arquitectura del bus de la AT a 32 bits y permite a más de una CPU comparti r el bus. EISA fue anunciado a fines de 1988 para competir con Microcanal de IBM. Las tarjetas de PC y AT existentes, que no pueden enchufarse en el Microcanal, sí pueden hacerlo en una ranura EISA.

Ethernet: Ethernet LAN estándar 802.3 de IEEE originalmente desarrollada por Xerox, Digital e Intel que utiliza el método de acceso CSMA/CD, transmite a 10Mbps y puede conectar en total hasta 1.024 nodos.

Ethernet estándar (También llamado Ethernet "thick" denso) utiliza una topología de bus con una longitud de cable con un máximo de 1.640 pasos sin utilizar un repetidor. La unión del cable se rea liza mediante una abrazadera que sitúa un emisor−receptor.

Ethernet estrecho (También llamado ThinNet y CheaperNet) es una topología de bus con una longitud de cable con un máximo de 607 pasos. Los nodos van encadenados por margarita con conectores BNC tipo T, mientra s que los emisores−receptores se encuentran dentro de las tarjetas adaptadoras de redes.

Ethernet de cable de dos hilos trenzados permite utilizar las líneas telefónicas instaladas (Si es el tipo adecuado) y Ethernet de fibra óptica es impenetrable por radiaciones externas. Ambos utilizan topología en estrella, lo cual se considera mucho más fácil de depurar cuando las redes se expandan.

FAT: (File Allocation Table). Tabla de Asignación (Distribución) de Archivos. La parte del sistema de archivos del DOS y OS/2 que lleva la cuenta de dónde están almacenados los datos en un disco. Es una tabla con una entrada para cada "cluster " en el disco, y la tabla completa está duplicada. El directorio, el cual contiene identificación del archivo (Nombre, extensión, fecha de última actualización, ...) apunta a las entradas de la FAT en donde comienzan los archivos. Si un archivo ocupa más de un "cluster", esa entrada apunta a otra entrada y así sucesivamente. Si un "cluster" se daña, su entrada correspondiente en la FAT se marca y no se usa nuevamente.

FDDI: (Fiber Distributed Data Interface). Interfaz de Distribución de Datos de Fibra Optica. Conjunto de

normas de ANSI para redes de área local con fibra óptica. Se aplica a las dos capas inferio res del modelo OSI (Enlace de datos y física) y transmite a 100 megabits por segundo. A esta velocidad, los gráficos de alta resolución pueden ser transmitidos rápidamente y el video digital puede ser manipulado en tiempo real.

FTP: (File Transfer Protocol). Protocolo de Transferencia de Archivos. Un protocolo TCP/IP que se usa para conectarse a la red, listar directorios y copiar archivos. También puede traducir entre ASCII y EBCDIC.

GUI: (Graphical User Interface). Interfaz Gráfica de Usuario. Basada en gráficos que incorpora iconos, menúes enrollables y un ratón, tal como se encuentra en los entornos de Macintosh, Windows, OS/2 Presentation Manager y GEM. Nótese la diferencia con interfaces de usuarios que se basan en caracteres o en texto, como el DOS, que presenta datos en el modo estándar de texto de 25 líneas y 80 columnas.

Hacker: La actividad de los hackers tiene que ver básicamente con el aprendizaje de todo lo que hay que conocer de un sistema, la posibilidad de sumergirse en éste al grado de abstraerse y la capacidad de repa rarlo cuando se cae. En general, a los hackers les interesa conocer el funcionamiento de los sistemas que encuentran interesantes. Aunque esas actividades no les atraen por cuestiones de dinero o por motivos de venganza, como lo hacen los crackers.

Hardware: Toda la maquinaria y el equipamiento. Compárese con software, el cual es un conjunto de instrucciones que le dicen a la computadora que hacer. También compárese con los datos, que son los hech os y cifras que se almacenan en el hardware y son controlados por el software.

En operación, una computadora es a la vez hardware y software. Uno es inútil sin el otro, y cada uno regula al otro. El diseño del hardware especifica que instrucciones puede seguir, y luego las instrucciones le dicen que hacer.

Tan inseparables como son el hardware y el software en operación, ellos son completamente diferentes cuando están siendo evaluados. El hardware es el mundo del almacenamiento y la transmisión. El software es e l mundo de la lógica y del lenguaje.

Cuanto más memoria y almacenamiento en disco tiene un sistema informático, más trabajo puede hacer. Cuanto más rápidos sean la memoria y los discos, para transmitir datos e instrucciones entre el los y la CPU, más rápido se hará el trabajo. Un problema de usuario puede ser traducido a un requerimiento de hardware basado en el tamaño de los archivos y las bases de datos que serán creadas, y el número de usuarios simultáneos en las terminales. El software, por otro lado, es más difícil de especificar. Los programas deben procesar adecuadamente las transacciones de negocios de la organización, e incluso la más peque&nt ilde;a compañía puede tener transacciones complicadas.

El hardware siempre trata el problema del procesamiento de datos del mismo modo. Â¿Cuánto?, Â¿Con qué rapidez? Pero el software se ocupa de los detalles tediosos de un negocio en constante cambio. Es mucho más difícil analizar, diseñar y desarrollar la solución de software que especificar el hardware.

HCSS: (High Capacity Storage Systems). Sistema de Almacenamiento de Alta Capacidad.

HPFS: (High Performance File System). Sistema de Archivos de Alto Rendimiento. Un sistema de archivos, presentado con OS/2 versión 1.2, que maneja discos más grandes (Volúmenes de 2TB; archivos de 2GB), nombr es largos de archivos (256 bytes) y puede lanzar el programa por referencia a los datos como en la Macintosh. Coexiste con el sistema FAT existente.

HTTP: (HyperText Transfer Protocol). Protocolo de Transferencia de Hipertexto.

RPC: (Remote Procedure Calls). Llamadas a Procedimientos Remotos.

ICMP: (Internet Control Message Protocol). Protocolo de Control de Mensajes de Internet.

IEEE: (Institute of Electrical and Electronic Engineers). Instituto de Ingenieros Eléctricos y Electrónicos.

IGMP: (Internet Group Management Protocol). Protocolo de Administración de Grupos de Internet.

Interactivo: Un diálogo bilateral entre el usuario y una computadora.

IPX/SPX: (Internetwork Packet Exchange / Sequenced Packet eXchange). Intercambio de Paquetes entre Redes / Intercambio Secuencial de Paquetes. IPX es un protocolo de comunicaciones del NetWare de Novell que se utiliza para encami nar mensajes de un nodo a otro. Los programas de aplicación que manipulan sus propias comunicaciones cliente/servidor o de igual a igual en una red Novell pueden acceder directamente al IPX o al protocolo SPX de NetWare. El IPX no garantiza la entr ega del mensaje. SPX es un protocolo de comunicaciones de Novell NetWare que se utiliza para comunicaciones entre procesos (Interprocess Communications − IPC). Garantiza que un mensaje completo llegue intacto, y emplea el protocolo NetWare IPX como mecani smo de distribución.

IRQ: (Interrupt ReQuest). Requerimiento de Interrupción. Una interrupción de hardware de un dispositivo periférico que demanda la atención del procesador.

ISDN: (Integrated Services Digital Network). Red Digital de Servicios Integrados. Un estándar internacional de telecomunicaciones para la transmisión de voz, video y datos a través de una línea de comu nicaciones digitales. Emplea la señalización Out−Of−Band, que provee un canal separado para la información de control. Los servicios ISDN se presentan en dos formas: (1) BRI (Basic Rate Interface − interfaz de régimen bá sico) y (2) PRI (Primary Rate Interface − interfaz de régimen primario).

ISO: (International Standards Organization). Organización Internacional de Estándares Una organización que establece estándares (Normas) internacionales, fundada en 1946, con sede en Ginebra. Se ocupa de todos los campos, excepto la electricidad y la electrónica, las cuales están ya desde antes bajo la jurisdicción de la IEC (International Electrotechnical Commission − Comisión Electrotécnica Internacional), tambi&eac ute;n radicada en Ginebra. Con respecto a los estándares de procesamiento de la información, la ISO y la IEC crearon la JTC1 (Joint Technical Committee − Comité Técnico Conjunto) para la tecnología informática.

La ISO desarrolla su trabajo a través de más de 160 comités técnicos y 2300 subcomités y grupos de trabajo, y está constituida por las organizaciones de estándares de más de 7 5 países, algunas de las cuales sirven como secretariados para estos cuerpos técnicos. En los EE.UU., la ANSI es miembro de la ISO.

Kerberos: Sistema de seguridad desarrollado en MIT que autentifica a los usuarios. No ofrece autorización de los servicios técnicos, ni de las bases de datos; establece identidad en la entrada al sistema, lo cual se utiliza en toda la sesión.

LAN: (Local Area Network). Red de Area Local.

Latencia: Estado latente. El tiempo entre la iniciación de un pedido de datos y el comienzo de la efectiva transferencia de los datos. En un medio rotativo, como un disco, es el tiempo que necesita para que el sector selec cionado gire y se coloque bajo el cabezal de lectura/escritura. El estado latente de un canal es el tiempo que tarda el canal de la computadora en desocuparse con objeto de transferir datos.

Login: Entrada de identificación, conexión. Igual que logon.

Mainframe: Macrocomputadora. Una computadora grande. A mediados de los años 60, todas las computadoras eran mainframes (Literalmente "bastidor principal"), ya que el término se refería al gabinete

que conten& iacute;a la CPU. Aunque mainframe aún significa gabinete principal, usualmente se refiere a un gran sistema de computación y toda la experiencia asociada que va con el.

Hay macrocomputadoras de escala pequeña, media y grande, manejando desde un manojo a varios miles de terminales en línea. Las macrocomputadoras de gran escala pueden tener centenares de megabytes de memoria principal y cen tenares de gigabytes de almacenamiento en disco. Las macrocomputadoras de media y gran escala usan computadoras más pequeñas como procesadores frontales que se conectan directamente a las redes de comunicaciones.

Los fabricantes originales de macrocomputadoras fueron Burroughs, Control Data, GE, Honeywell, IBM, NCR, RCA y Univac, conocidos también como IBM y los siete enanitos. Después de que las divisiones de computadoras de GE y RCA fueran absorbidas por Honeywell y Univac respectivamente, los fabricantes de macrocomputadoras fueron conocidos como IBM.

Multiprocesamiento: El procesamiento simultáneo con dos o más procesadores en una computadora, o dos o más computadoras que estén procesando juntas. Cuando se usan dos o más computadoras, se lig an con un canal de alta velocidad y comparten la carga de trabajo general entre ellas. En caso de que una deje de operar, la otra se hace cargo. En sistemas con tolerancia de fallos, dos o más procesadores se construyen dentro de un mismo gabinete.

El multiprocesamiento también se efectúa en computadoras de propósitos especiales, como los procesadores matriciales, los cuales proveen procesamiento matemático simultáneo de conjuntos de datos.

A pesar de que las computadoras se construyen con diversas características que se superponen, como ejecutar instrucciones mientras se introducen y sacan datos, el multiprocesamiento se refiere específicamente a ejecuci&oac ute;n de instrucciones simultáneas.

Multitarea: La ejecución de dos o más programas en una computadora al mismo tiempo. La multitarea se controla por el sistema operativo, que carga los programas y los maneja hasta que terminen. El número de pr ogramas que pueden ser efectivamente ejecutados depende de la cantidad de memoria disponible, la velocidad de CPU, capacidad y velocidad de los recursos periféricos, así como también de la eficiencia del sistema operativo.

La multitarea se realiza debido a las diferencias de velocidades de entrada/salida y procesamiento. Mientras un programa está esperando una entrada, se pueden ejecutar instrucciones de otro programa. Con programas interacti vos, los segundos de demora entre entradas de teclado se usan para ejecutar instrucciones de otros programas. En sistemas de procesamiento por lotes, los milisegundos de demora entre entrada y salida de datos de la computadora se usan para ejecutar instru cciones en otros programas.

Tradicionalmente, multitarea significaba ejecutar dos o más tareas dentro del mismo programa al mismo tiempo, y multiprogramación significaba ejecutar dos o más programas en la computadora al mismo tiempo. Hoy, mult itarea significa multiprogramación y multilectura significa multitarea.

NCP: (1) (Network Control Program). Programa para Control de Redes. Un programa que controla el tráfico entre múltiples terminales y una mini o macrocomputadora. Típicamente reside en un procesador frontal y ejecuta operaciones tales como el censo de las terminales. (2) (NetWare Core Protocol) Lenguaje patentado usado en las redes NetWare de Novell para la comunicación entre la estación de trabajo y el servidor.

NDIS: (Network Driver Interface Specification). Especificación de Interfaces para Controladores de Red. Una especificación de Microsoft para escribir controladores independientes del hardware en el estrato de enlace s de datos (Método de acceso a los medios). Cuando los protocolos de transporte se comunican con la especificación NDIS, las tarjetas de red con controladores MAC obedientes al NDIS pueden ser libremente

intercambiadas.

NDS: (Novell NetWare Directory Services). Servicio de Directorios de Novell NetWare.

NetBEUI: (NETBIOS Extended User Interface). Interfaz de Usuario Extendido de NetBIOS. La realización del protocolo de transporte NetBIOS en LAN Manager y LAN Server. Se comunica con las tarjetas de interfaz de red (NICs) v ía NDIS (Network Driver Interface Specification).

El término fue originalmente usado para definir el protocolo NetBIOS después que este fue mejorado para soportar la Token Ring Network.

NetBIOS: Es un protocolo de transporte comúnmente usado para redes de área local de PC introducido con la Red para PC de IBM e implementado en MS−Net de Microsoft y LAN Manager. Los programas de aplicaci&oacut e;n usan NetBIOS para comunicaciones cliente/servidor o de igual a igual.

Hay dos modos NetBIOS para comunicaciones. El Datagrama es el método más rápido, pero no garantiza la entrega de un mensaje. Es un paquete independiente, con el nombre del remitente y del destinatario, usualmente li mitado a 512 bytes. Si el dispositivo receptor no está atento a los mensajes, el datagrama se pierde. El modo Session establece una conexión hasta ser interrumpida. Garantiza la entrega de mensajes de hasta 64K bytes. Los protocolos acordes con NetBIOS se refieren a los estratos 3, 4 y 5 en el modelo OSI.

NFS: (Network File System). Sistema de Archivos de Red.

NIC: (Network Interface Card). Tarjeta de Interfaz de Redes.

NLM: (NetWare Loadable Module). Módulos Cargables de NetWare.

Nodo: En comunicaciones, un punto de empalme o conexión en una red (Una terminal o una computadora).

NOS: (Network Operating System). Sistema Operativo de Red.

OSI: (Open System Interconnection). Interconexión de Sistemas Abiertos. Un modelo de referencia que fue definido por la ISO (International Standards Organization) como un estándar para las comunicaciones mundiales. Define una estructura para implementación de protocolos en siete estratos o capas.

Password: Contraseña, palabra de paso. Palabra o código utilizado para identificar a un usuario autorizado; es normalmente provisto por el sistema operativo o manejadores de bases de datos (DBMS). Las contrase&ntild e;as sirven como una medida de seguridad contra el acceso no autorizado a los datos; de todos modos, la computadora sólo puede verificar la legitimidad de la contraseña y no la legitimidad del usuario.

Protocolo: En comunicaciones, un conjunto de normas y regulaciones que gobiernan la transmisión y recepción de datos.

Puente (Bridge): Un dispositivo que conecta dos redes de igual tipo. Contrástese con gateway, que interconecta dos diferentes tipos de redes.

RAM: (Random Access Memory). Memoria de Acceso Aleatorio. Es el almacenamiento de trabajo de la computadora, que físicamente es una colección de chips. Es un recurso importante de la computadora, ya que determina el tamaño y el número de programas que pueden ejecutarse al mismo tiempo, como también la cantidad de datos que pueden ser procesados instantáneamente.

RFC: (Request For Comments) Peticiones Para Comentarios de Internet. Las peticiones para comentarios

proporcionan la documentación oficial, las políticas y los procedimientos de la comunidad de Internet.

Ruteador (Router): Encaminador, director. En comunicaciones, dispositivo que selecciona un recorrido de viaje adecuado, y encamina un mensaje de acuerdo a el. Los encaminadores se emplean en redes complejas en las que hay m&uacut e;ltiples vías de comunicación entre los usuarios de la red. El encaminador examina la dirección de destino del mensaje y determina la ruta más efectiva.

SIMM: (Single In−line Memory Module). Chips de memoria montados sobre una pequeña tabla de baquelita con sus conectores dispuestos sobre un solo lado de la tarjeta.

SMB: (Server Message Block). Bloque de Mensajes de Servidor. Formato de mensajes usado en el protocolo de archivos compartido Microsoft/3Com para PC Network, MS−Net y LAN Manager. Se usa para transferir solicitudes de archivos en tre estaciones de trabajo y servidores, y también dentro de servidor para operaciones internas. Cuando se transfieren a través de la red, los SMB son transportados dentro del paquete "NetBIOS Network Control Block" (NCB − bloque de control d e red).

Software: Instrucciones para una computadora. Una serie de instrucciones que realizan una tarea en particular se llama programa o programa de software. Las dos categorías principales son software de sistemas y software de aplicaciones.

El software de sistemas se compone de programas de control, incluyendo el sistema operativo, software de comunicaciones y administrador de bases de datos.

El software de aplicaciones es cualquier programa que procesa datos para el usuario (Inventario, nómina, hoja de cálculo, procesador de texto, etc.).

SQL: (Structured Query Language). Lenguaje Estructurado de Consulta. Lenguaje utilizado para interrogar y procesar datos en una base de datos relacional. Desarrollado originalmente por IBM para sus macrocomputadoras, han habido m uchas implementaciones creadas para aplicaciones de base de datos en mini y microcomputadoras. Las órdenes (Mandatos) de SQL se pueden utilizar para trabajar interactivamente con una base de datos, o pueden incluirse en un lenguaje de programaci&oa cute;n para servir de interfaz a una base de datos.

TCP/IP: (Transmission Control Protocol /Internet Protocol). Protocolo de Control de Transmisión / Protocolo de Internet.

Telnet: Protocolo de emulación de terminales desarrollado originariamente por ARPAnet.

TPDU: Unidad de datos del protocolo de transporte.

UNA: (Universal Networking Architecture). Arquitectura Universal de Red.

UPMS: (User Profile Management Services). Administración de servicios de perfiles de usuario.

VAP: (Value Added Process). Proceso de Valor Añadido. Un programa que mejora o proporciona funciones adicionales de servidor a un servidor de NetWare 286. El soporte para diferentes clases de estaciones de trabajo, m&aacut e;quinas de bases de datos, servidores de fax y de impresión son ejemplos. En NetWare 386, el VAP se llama NetWare Loadable Module (NLM − módulo cargable NetWare).

VPN.−(Virtual Private Network). Red Virtual Privada.

B I B L I O G R A F I A

ANDRES S. TANENBAUM. Sistemas Operativos Modernos. 1992.

JOSE CAÑETE V. Microsoft Windows NT. 1997.

Al SERVATI La Biblia de Intranet. 1997.

ERICK JIMENES M. Sistemas Operativos de Redes.

www.tecmor.mx/mats−dsc/sor/indice"1.html

JOSE GONZALES M. Implementación de una Intranet sobre Windows NT.

www.globalnet.com.mx/intranet/index.html

MICROSOFT Documento NT4UNIX.DOC.

www.microsoft.com

Arquitectura del Sistema Operativo Windows NT Página: 63

- Apoyo de nombres largos de archivos
- Apoyo de seguridad local.
- Tamaño máximo de partición:

16 Exabytes (teóricamente)

2 Terabytes (actual).

- Apoyo de nombres largos de archivos
- Ninguna seguridad local.
- Tamaño máximo de partición:

4 Gb.

N T F S

F A T

PROPIEDADES

PROPIEDADES