PARTE IV

ESTUDOS DE CASOS

"Quem abre uma escola, fecha uma prisão."

VICTOR HUGO (1802-1885)
Escritor e poeta francês

"A leitura faz o homem completo.
A história torna-o sábio e prudente;
a poesia, espiritual;
a matemática, sutil;
a filosofia, profundo;
a moral, grave;
a lógica e a retórica, apto para discutir.
Ler é conversar com os sábios."

FRANCIS BACON (1561-1626)
Filósofo inglês

"Onde há educação não há distinção de classe."

CONFÚCIO (550 a.C.-479 a.C.)
Filósofo chinês

14

MICROSOFT WINDOWS

14.1 Histórico

A Microsoft lançou em 1981 seu primeiro sistema operacional, o MS-DOS (Disk Operating System), para a linha de computadores pessoais IBM-PC, concebido para ser um sistema operacional de 16 bits, monoprogramável, monousuário e com uma interface de linha de comando. O MS-DOS foi desenvolvido com base no sistema operacional CP/M e algumas idéias do Unix.

Em 1985, é lançada a primeira versão do MS Windows, que introduziu uma interface gráfica, porém manteve o MS-DOS como o sistema operacional. As versões posteriores do MS Windows, como Windows 3.0, Windows 95, Windows 98 e Windows Me, apesar de várias melhorias e inovações, sempre tinham o MS-DOS como o núcleo do sistema operacional.

Devido às inúmeras limitações e deficiências do MS-DOS, a Microsoft começou a conceber um novo sistema operacional em 1988, conhecido como Windows NT (New Technology). Este novo projeto foi conduzido por David Cutler, ex-projetista da Digital Equipment Corporation (DEC), que foi responsável pelo desenvovimento de inúmeros sistemas operacionais, como o PDP/RSX e o VAX/VMS. Além da grande influência do sistema operacional VMS, no projeto do Windows NT foram utilizados vários conceitos dos sistemas OS/2 e LAN Manager.

Em 1993, a Microsoft lança o Windows NT nas versões para desktop e servidores, sistema operacional de 32 bits, com multitarefa preemptiva, multithread, memória virtual e suporte a múltiplos processadores simétricos. O Windows NT não tem qualquer relação com a arquitetura do MS-DOS, mas oferece compatibilidade parcial com aplicações legadas, além de ter incorporado algumas de suas características, como a interface gráfica. Com isso, passaram a existir duas linhas de sistemas operacionais com arquiteturas completamente distintas, porém com a mesma interface para o usuário (Fig. 14.1).

O Windows 2000 é uma evolução do Windows NT versão 4, pois mantém a mesma arquitetura interna. O Windows 2000 passou a incorporar alguns recursos da família

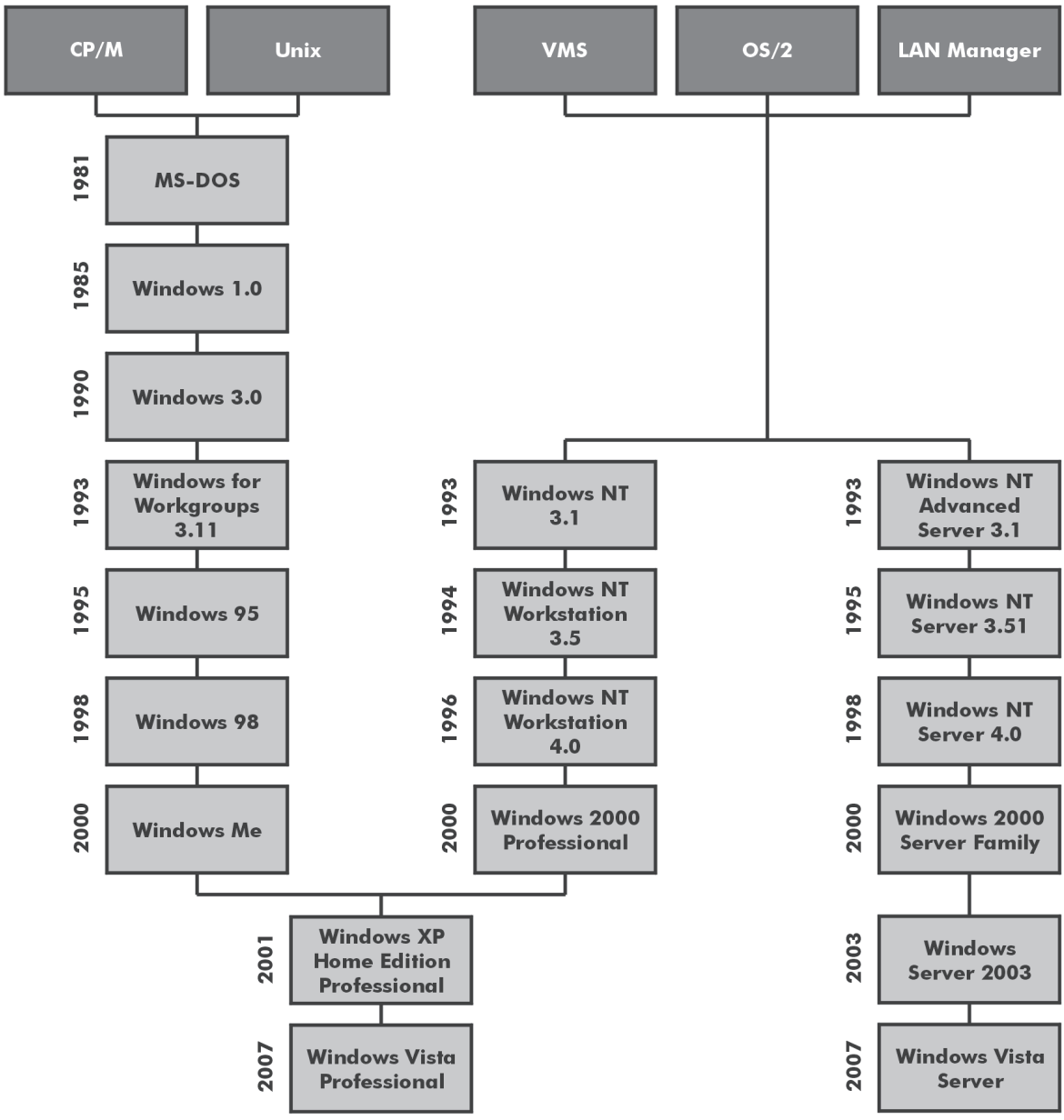

Fig. 14.1 Histórico dos sistemas operacionais MS Windows.

DOS-Windows, como a função de plug-and-play. A grande novidade trazida pelo sistema é o Active Directory, que funciona como um serviço de diretórios e veio substituir o conceito de domínio existente no Windows NT.

O Windows XP, lançado em 2001, introduz uma nova interface gráfica e alguns poucos recursos adicionais, porém mantém a mesma arquitetura do Windows 2000. A partir do Windows XP, a Microsoft começou a descontinuar as famílias DOS-Windows e Windows NT/2000, integrando as duas linhas de sistemas operacionais. Em 2003, é lançado o Windows Server 2003, nova versão do Windows 2000 Server, com suporte a processadores de 64 bits.

Para 2007/2008, a Microsoft está lançando o Windows Vista para desktops e servidores. Sua expectativa é que esta nova versão do MS Windows seja o sistema operacional utilizado em grande escala na década de 2010.

14.2 Características

Este capítulo apresenta as características do Windows Server 2003, sistema operacional multiprogramável de 32/64 bits, que suporta escalonamento preemptivo, multithread, multiusuário, multiprocessamento simétrico (SMP) e gerência de memória virtual.

O Windows Server 2003 apresenta seis versões, que possuem a mesma arquitetura interna e interface gráfica. A diferença entre as versões está no suporte ao número de processadores, tamanho da memória física, número de conexões de rede e serviços oferecidos. A Tabela 14.1 apresenta uma comparação entre as principais características das diferentes versões do Windows Server 2003.

Tabela 14.1 Versões do Windows Server 2003

Versão	Máximo UCPs (32 bits)	Máximo Memória (32 bits)	Máximo UCPs (64 bits)	Máximo Memória (64 bits)
Windows Server 2003 Web Edition	2	2 GB	–	–
Windows Server 2003 Small Business Server	2	2 GB	–	–
Windows Server 2003 Standard Edition	4	4 GB	–	–
Windows Server 2003 Enterprise Edition	8	32 GB	8	64 GB
Windows Server 2003 Datacenter Edition	32	128 GB	64	1024 GB

14.3 Estrutura do Sistema

O MS Windows possui mais de 40 milhões de linhas de código escritas, em sua maioria, em Linguagem C, porém com alguns módulos desenvolvidos em C++ e assembly. O sistema é estruturado combinando o modelo de camadas e o modelo cliente-servidor. Embora não seja totalmente orientado a objetos, o MS Windows representa seus recursos internos como objetos. Essa abordagem diminui o impacto das mudanças que o sistema venha a sofrer no futuro, reduzindo o tempo e o custo de manutenção. Além disso, criação, manipulação, proteção e compartilhamento de recursos podem ser feitos de forma simples e uniforme. Quando um objeto é criado, um handle é associado a ele, permitindo o seu acesso e compartilhamento. A Tabela 14.2 apresenta alguns objetos implementados pelo sistema.

Tabela 14.2 Objetos do Windows 2003

Objetivo	Descrição
Processo	Ambiente para execução de um programa.
Thread	Unidade de execução em um processo.
Seção	Área compartilhada de memória.
Porta	Mecanismo para a troca de mensagens entre processos.
Token de acesso	Identificação de um objeto.
Evento	Mecanismo de sincronização entre processos.
Semáforo	Contador utilizado na sincronização entre processos.
Timer	Temporizador.

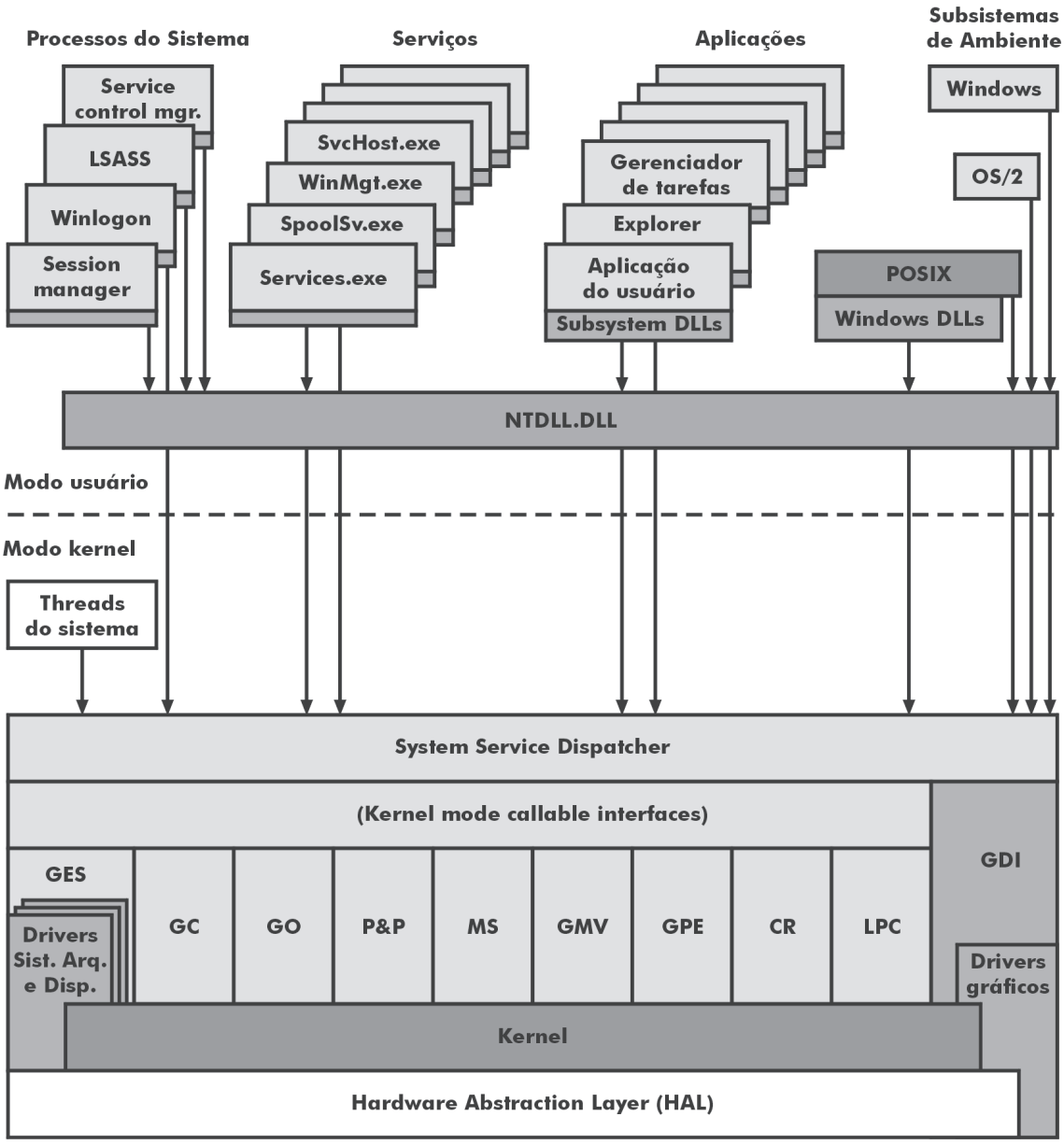

Fig. 14.2 Arquitetura do sistema.

A arquitetura do MS Windows pode ser dividida em duas camadas, como pode ser observado na Figura 14.2. No modo usuário estão os processos do sistema, serviços, aplicações, subsistemas de ambiente e subsistemas de DLLs. No modo kernel está o núcleo do sistema propriamente dito e o HAL.

- Hardware Abstraction Layer

  O Hardware Abstraction Layer (HAL) é uma biblioteca que engloba a parte do código do sistema dependente do hardware, como acesso a registradores e endereçamento de dispositivos, identificação de interrupções, DMA, temporização, sincronização em ambientes com múltiplos processadores e interface com a BIOS e CMOS. Essa camada garante ao Windows uma facilidade muito grande de ser portado para diferentes plataformas de hardware.

- Kernel

  O kernel é responsável por tornar o sistema operacional totalmente independente do hardware, implementado as demais funções não suportadas pelo HAL, como troca de contexto, escalonamento e dispatching, tratamento de interrupções e exceções, e suporte a objetos de uso interno do kernel, como DPC (Deferred Procedure Call), APC (Asynchronous Procedure Call) e dispatcher objects.

- Device Drivers

  No MS Windows, um device driver não manipula diretamente um dispositivo. Esta tarefa é função do HAL, que serve de interface entre o driver e o hardware. Esse modelo permite que a maioria dos device drivers seja escrita em Linguagem C e, conseqüentemente, tenha grande portabilidade.

- Executivo

  O executivo é a parte do núcleo totalmente independente do hardware. Suas rotinas podem ser agrupadas em categorias, em função do tipo de serviço que oferecem, como gerência de objetos (GO), gerência de E/S (GES), gerência de processos e threads (GPE), gerência de memória virtual (GMV), monitor de segurança (MS), gerência de cache (GC), gerenciamento de energia e plug-and-play (P&P), configuração do registry (CR), Local Procedure Call (LPC) e Graphics Device Interface (GDI).

- NTDLL.DLL

  O NTDLL é uma biblioteca do sistema que faz a interface entre as aplicações que utilizam o subsistema de DLLs e o executivo. Uma DLL (Dynamic Link Libraries) é uma biblioteca de procedimentos, geralmente relacionados, que são linkados à aplicação apenas em tempo de execução. Esse tipo de biblioteca compartilhada evita que aplicações que utilizem os mesmos procedimentos tenham sua própria cópia invidual das rotinas na memória principal.

- Subsistemas

  O MS Windows oferece três subsistemas de ambiente: Windows, OS/2 e POSIX, que permitem que aplicações desenvolvidas nestes ambientes sejam suportadas pelo mesmo sistema operacional. Enquanto os subsistemas OS/2 e POSIX são opcionais, o subsistema Windows é obrigatório e deve estar sempre carregado.

  As aplicações que realizam chamadas às rotinas do sistema utilizam uma biblioteca, conhecida como Application Program Interface (API). Uma aplicação no MS Windows não pode realizar uma chamada diretamente a uma rotina do sistema utilizando uma system call, mas sempre fazendo uso de uma API. As APIs funcionam como uma interface entre a aplicação e as system calls propriamente ditas, que por sua vez fazem as chamadas às rotinas do sistema operacional. As APIs são implementadas utilizando o subsistema de DLLs.

14.4 Processos e Threads

Processos no MS Windows são objetos criados e eliminados pelo gerente de objetos, representados internamente por uma estrutura chamada EPROCESS (Executive Process Block). Um processo ao ser criado possui, dentre outros recursos do sistema, um espaço de endereçamento virtual, uma tabela de objetos, um token de acesso e pelo menos um thread (Fig. 14.3).

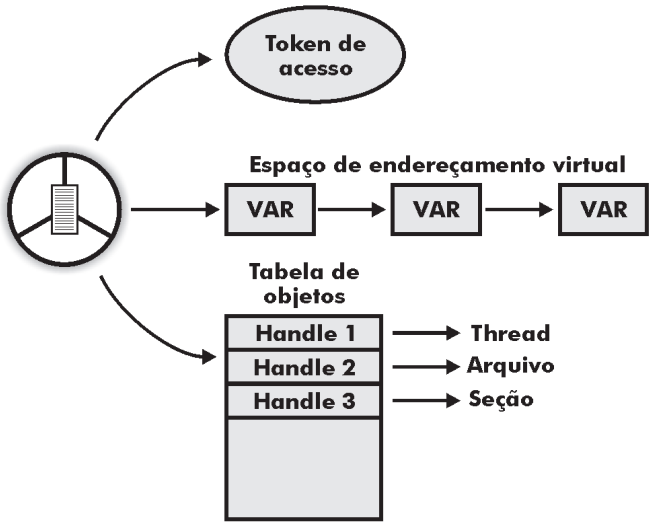

Fig. 14.3 Estrutura do processo.

O espaço de endereçamento virtual é formado por descritores, conhecidos como VARs (Virtual Address Descriptors), que permitem que a alocação do espaço de endereçamento seja feita dinamicamente, conforme as referências aos endereços virtuais. A tabela de objetos contém handles para cada objeto a que o processo tem acesso. Um handle é criado quando o processo cria um novo objeto ou solicita acesso a um já existente. O token de acesso identifica o processo para o sistema operacional, e sempre que o processo tem acesso a um recurso do sistema o token é utilizado para determinar se o acesso é permitido. Cada processo é criado com um único thread, mas threads adicionais podem ser criados e eliminados quando necessário.

Threads também são implementados como objetos, sendo criados e eliminados pelo gerenciador de objetos e representados por uma estrutura chamada ETHREAD (Executive Thread Block). Todos os threads de um processo compartilham o mesmo espaço de endereçamento virtual e todos os recursos do processo, incluindo token de acesso, prioridade-base, tabela de objetos e seus handles.

O MS Windows escalona apenas threads para execução, e não processos. A Tabela 14.3 apresenta os diferentes estados possíveis de um thread, enquanto a Fig. 14.4 apresenta as mudanças de estado de um thread durante sua existência.

Tabela 14.3 Estados de um thread

Estado	Descrição
Criação	Criação e inicialização do thread.
Espera	O thread aguarda por algum evento, como uma operação de E/S.
Execução	O thread está sendo executado.
Pronto	O thread aguarda para ser executado.
Standby	O thread foi escalonado e aguarda pela troca de contexto para ser executado. Somente um thread pode estar no estado de standby para cada processador existente.
Terminado	Quando um thread termina sua execução, o sistema o coloca no estado de terminado, porém o objeto pode ou não ser eliminado.
Transição	O thread aguarda que suas páginas gravadas em disco sejam lidas para a memória principal.

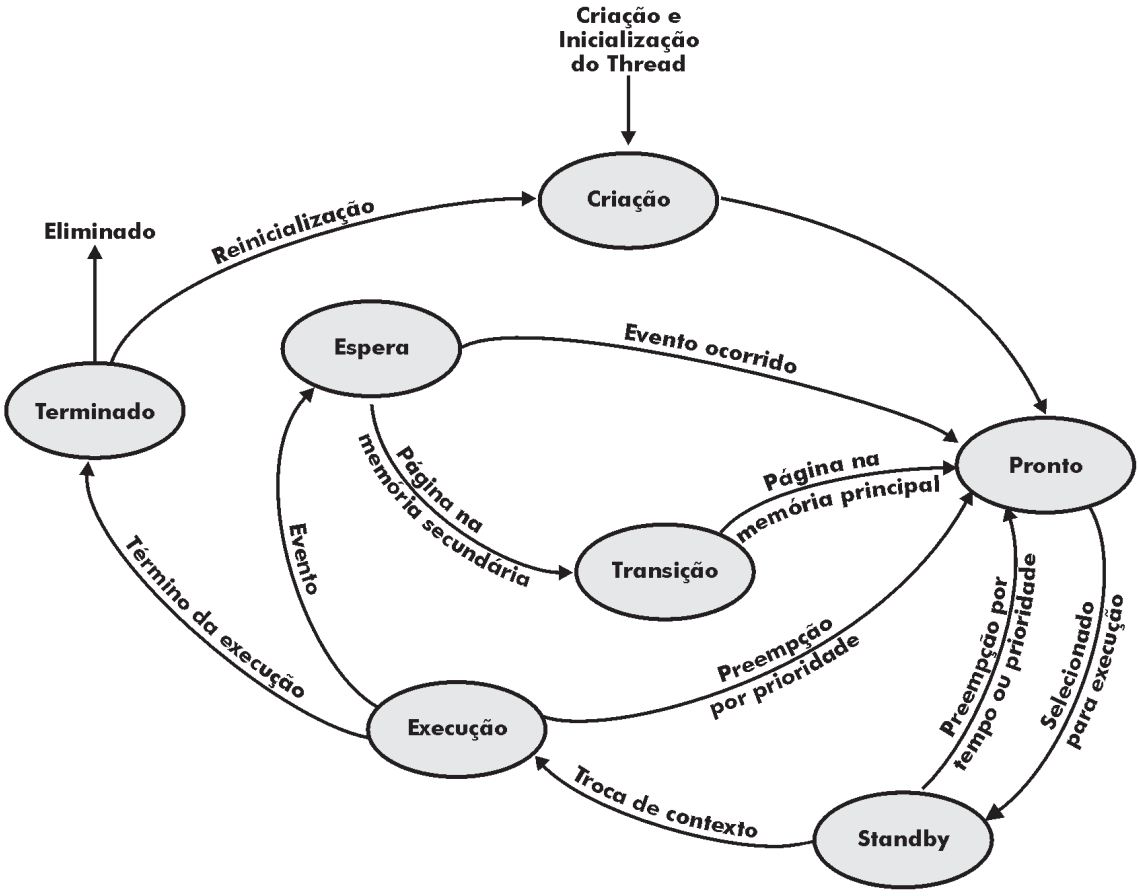

Fig. 14.4 Mudanças de estado de um thread.

Além de processos e threads, o MS Windows implementa os conceitos de job e fiber. Um job é uma coleção de processos que guardam alguma relação e devem ser gerenciados como uma unidade. Processos em um job compartilham quotas e privilégios, como o número máximo de processos que podem ser criados e o espaço máximo alocado na memória principal.

O MS Windows implementa threads em modos usuário e kernel. Threads em modo kernel, denominados simplesmente threads, apresentam problemas de desempenho devido à necessidade de troca de modo de acesso usuário-kernel-usuário. Threads em modo usuário, denominados fibers, eliminam as trocas de contexto e de modo de acesso e, conseqüentemente, oferecem melhor desempenho. Cada thread pode ter múltiplos fibers, da mesma forma que um processo pode ter múltiplos threads. O sistema operacional desconhece a existência dos fibers, ficando a cargo da própria aplicação o seu escalonamento.

A Tabela 14.4 apresenta algumas APIs utilizadas para criação, eliminação, comunicação e sincronização de processos, threads e fibers.

14.5 Gerência do Processador

O MS Windows suporta dois tipos de política de escalonamento: escalonamento circular com prioridades e escalonamento por prioridades. A política de escalonamento é

Tabela 14.4 A APIs para a gerência de processos, threads e fibers

API	Descrição
CreateProcess	Cria um processo.
CreateThread	Cria um thread no processo corrente.
CreateRemoteThread	Cria um thread em um outro processo.
CreateFiber	Cria um fiber no processo corrente.
OpenProcess	Retorna um handle para um determinado processo.
GetCurrentProcessID	Retorna a identificação do processo corrente.
ExitProcess	Finaliza o processo corrente e todos seus threads.
TerminateProcess	Termina um processo.
ExitThread	Finaliza o thread corrente.
TerminateThread	Termina um thread.
CreateSemaphore	Cria um semáforo.
OpenSemaphore	Retorna um handle para um determinado semáforo.
WaitForSingleObject	Espera que um único objeto, como um semáforo, seja sinalizado.
WaitForMultipleObjects	Espera que um conjunto de objetos sejam sinalizados.
EnterCriticalSection	Sinaliza que a região crítica está sendo executada.
LeaveCriticalSection	Sinaliza que a região crítica não está mais sendo executada.

implementada associando prioridades aos processos e threads. Inicialmente, o thread recebe a prioridade do processo ao qual pertence, podendo ser alterada posteriormente. O escalonamento é realizado considerando apenas os threads no estado de pronto, sendo que os processos servem apenas como unidade de alocação de recursos.

O MS Windows implementa 32 níveis de prioridades, divididos em duas faixas: escalonamento de prioridade variável (1-15) e de tempo real (16-31). A prioridade zero serve apenas para que um thread especial do sistema (zero page) seja executado quando não existirem threads prontos para execução. Threads com a mesma prioridade são organizados em filas no esquema FIFO, e o primeiro thread na fila de maior prioridade será sempre selecionado para execução. Caso um thread fique pronto com prioridade maior que o thread em execução, ocorrerá a preempção por prioridade (Fig. 14.5).

A seguir, são apresentados os dois esquemas de escalonamento implementado pelo Windows:

- Prioridade Variável

  No esquema de prioridade variável (1-15) é adotada a política de escalonamento circular com prioridades. O thread de maior prioridade é selecionado e recebe uma fatia de tempo para ser executado. Terminado este *quantum*, o thread sofre preempção por tempo e retorna para o estado de pronto no final da fila associada a sua prioridade.

  O escalonamento de prioridade variável utiliza dois tipos de prioridades para cada thread: base e dinâmica. A prioridade-base é normalmente igual ao do processo e não sofre alteração durante a existência do thread. Por outro lado, a prioridade dinâ-

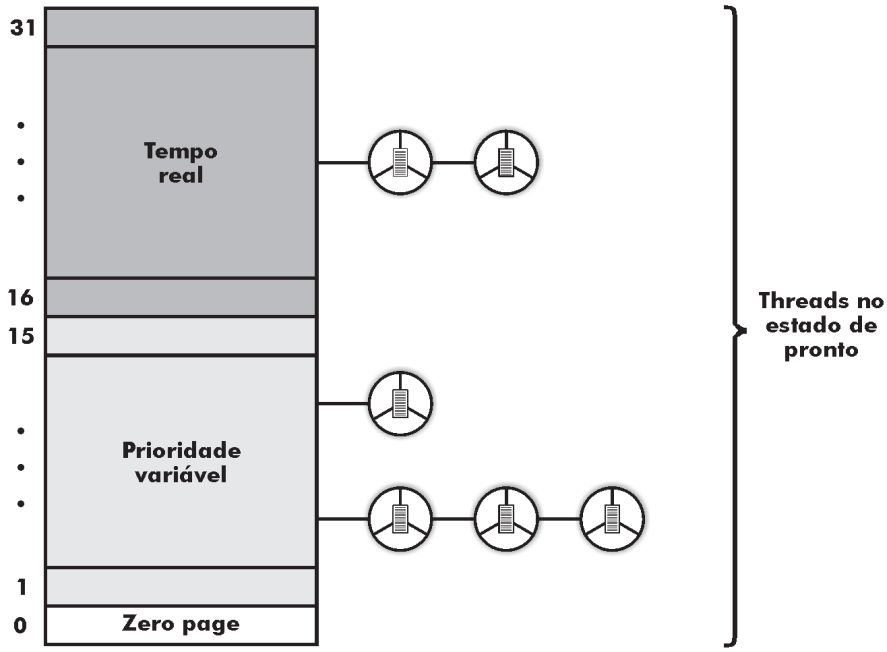

Fig. 14.5 Níveis de prioridade.

mica é sempre igual ou maior que a prioridade-base e varia de acordo com as características do thread, sendo controlada pelo sistema. O escalonamento é realizado em função da prioridade dinâmica dos threads.

A prioridade dinâmica é aplicada exclusivamente aos threads que passam pelo estado de espera. Por exemplo, sempre que um thread realiza uma operação de E/S o sistema o coloca no estado de espera. No momento em que o thread é transferido para o estado de pronto, a prioridade-base do thread é acrescida de um valor em função do tipo de operação realizado, ou seja, a prioridade dinâmica é resultado da soma da prioridade-base mais um incremento (Tabela 14.5). O incremento não é cumulativo, e a prioridade dinâmica nunca poderá ultrapassar o limite máximo da prioridade variável, ou seja, prioridade 15.

Tabela 14.5 Incremento na prioridade-base

Tipo de Operação de E/S	Incremento
Disco, CD e vídeo	1
Rede	2
Teclado e mouse	6
Som	8

O mecanismo de prioridade dinâmica permite equilibrar o uso do processador entre os diversos threads no sistema. Por exemplo, suponha que existam apenas dois threads no sistema com a mesma prioridade, um CPU-bound e outro I/O-bound. Toda vez que o thread I/O-bound realizar uma operação de E/S e retornar ao estado de pronto, o sistema incrementará a sua prioridade-base, fazendo com que o thread CPU-

bound sofra uma preempção. Dessa forma, enquanto o thread I/O-bound executa várias vezes, utilizando apenas parte da sua fatia de tempo, o thread CPU-bound executa menos vezes, porém utilizando todo o seu *quantum*.

- Tempo Real

  No esquema de tempo real (16-31) é adotada a política de escalonamento por prioridades. Existem duas grandes diferenças no esquema de tempo real, se comparado ao de prioridade variável: o conceito de fatia de tempo e a prioridade dinâmica.

  Os threads no estado de execução são processados o tempo que for necessário, não existindo fatia de tempo. Nesse caso, um thread é interrompido apenas por processos mais prioritários, ou seja, quando sofre uma preempção por prioridade. Além disso, os threads não têm aumentos de prioridade, sendo as prioridades-base e dinâmicas sempre iguais.

  Esta faixa de prioridade deve ser utilizada apenas por threads do sistema operacional por questões de segurança. Se qualquer thread com prioridade acima de 15 entrar em loop, nenhum dos threads com prioridades entre 1 e 15 será executado.

Os dois níveis de escalonamento apresentados permitem ao MS Windows oferecer características de sistemas de tempo compartilhado e tempo real dentro do mesmo ambiente, tornado-o versátil e possibilintado sua utilização por diferentes tipos de aplicações. A Tabela 14.6 apresenta algumas APIs relacionadas ao escalonamento de threads.

Tabela 14.6 APIs de escalonamento

API	Descrição
SuspendThread	Coloca o thread no estado de espera.
ResumeThread	Permite que o thread volte para o estado de pronto.
SetPriorityClass	Permite alterar a prioridade-base do processo.
SetThreadPriority	Permite alterar a prioridade-base do thread.
SetThreadPriorityBoost	Permite oferecer um incremento de prioridade temporário ao thread.
Sleep	Coloca o thread em estado de espera por um certo intervalo de tempo.

14.6 Gerência de Memória

O MS Windows implementa o mecanismo de gerência de memória virtual por paginação. A gerência de memória, ao contrário do escalonamento, lida com processos e não threads, ou seja, o espaço de endereçamento virtual pertence ao processo. O tamanho do espaço de endereçamento virtual depende da arquitetura de hardware. Em máquinas com 32 bits, cada processo pode endereçar 4 Gb, sendo 2 Gb reservados para o sistema operacional e 2 Gb para aplicações do usuário. Cada processo tem o seu espaço de endereçamento virtual único, independente dos demais processos, enquanto o espaço de endereçamento do sistema é compartilhado por todos os processos. A Fig. 14.6 apresenta o espaço de endereçamento virtual em uma arquitetura de 32 bits.

No MS Windows, o tamanho da página é definido em função da arquitetura do processador, possibilitando páginas entre 4 Kb (32 bits) e 2 Mb (64 bits). No processador Pentium, da Intel, as páginas possuem 4 Kb. Em arquiteturas de 32 bits, o mapeamento é realizado utilizando tabelas de páginas em dois níveis, sendo que a tabela de primei-

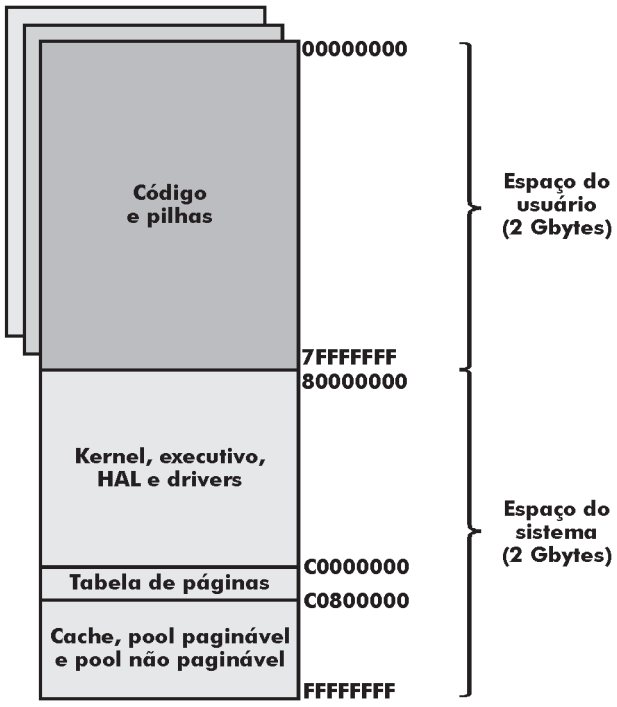

Fig. 14.6 Espaço de endereçamento virtual em uma arquitetura de 32 bits.

ro nível (page directory index) aponta para tabelas de segundo nível (page table index) que, por sua vez, apontam para os frames na memória principal (Fig. 14.7). O endereço real é obtido combinando-se o endereço do frame e o deslocamento. Em arquiteturas de 64 bits, o MS Windows utiliza um esquema de mapeamento em quatro níveis, ou seja, utilizando quatro tabelas de endereçamento.

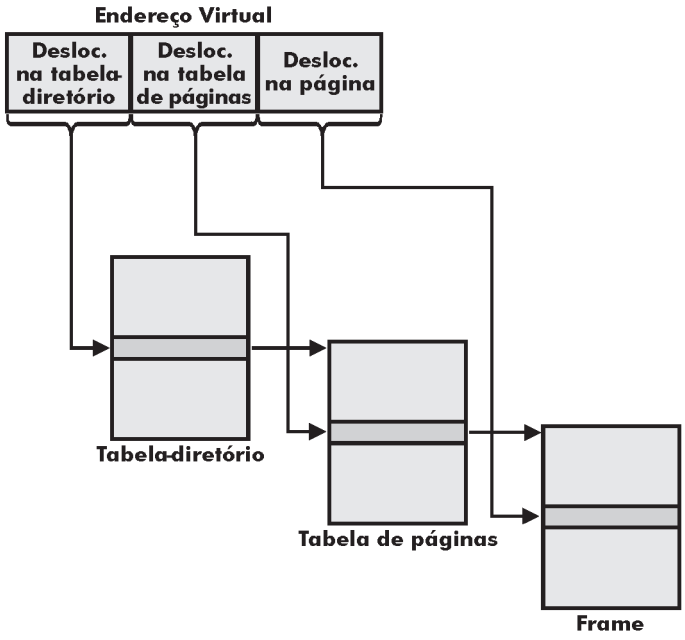

Fig. 14.7 Mapeamento.

Cada página da memória virtual gera uma entrada na tabela de páginas denominada *page table entry* (PTE). O PTE possui, entre outras informações, bits que indicam a localização da página, um bit que indica se a página foi alterada (*dirty bit*) e bits que permitem implementar a proteção da página.

A gerência de memória permite que dois ou mais processos compartilhem a mesma área de código ou dados na memória principal. Não existe nenhum mecanismo automático de controle de acesso a essa região, ficando como responsabilidade de cada processo a sincronização para evitar conflitos. Além disso, é possível mapear um arquivo residente em disco na memória principal e compartilhá-lo entre diversos processos. As aplicações podem ler e gravar registros diretamente na memória principal, evitando operações de E/S em disco.

O MS Windows implementa o esquema de paginação por demanda como política de busca de páginas. Quando ocorre um *page fault*, o sistema carrega para a memória principal a página referenciada e um pequeno conjunto de páginas próximas (cluster de páginas), na tentativa de reduzir o número de operações de leitura em disco e melhorar o desempenho do sistema (Fig. 14.8a). O tamanho do cluster de páginas varia conforme o tamanho da memória principal e se a página lida for de código ou dados.

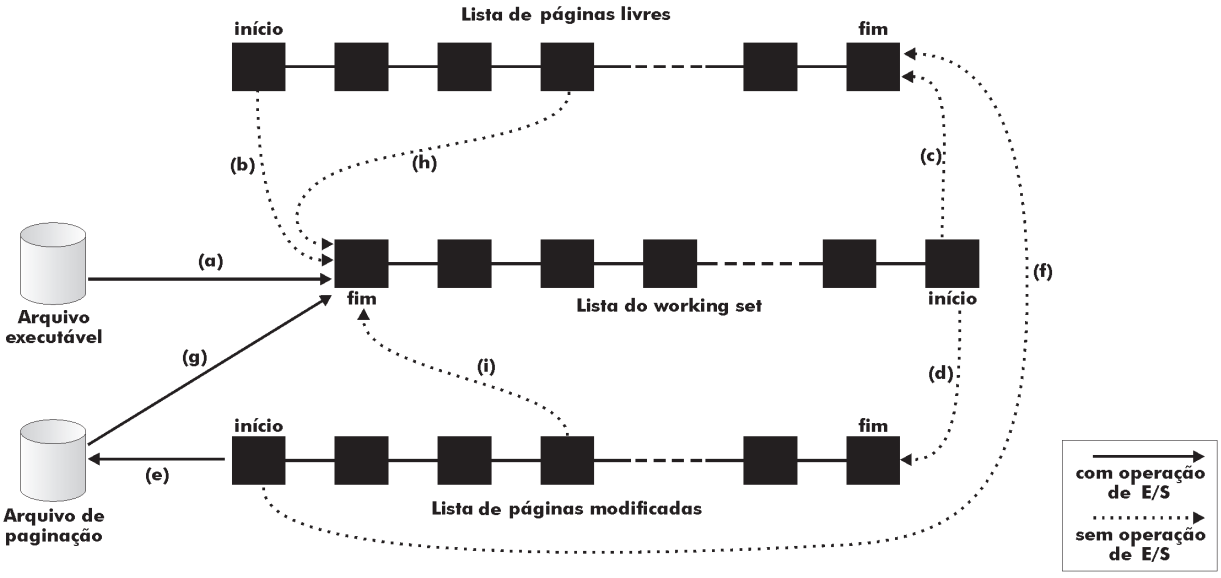

Fig. 14.8 Gerência de memória.

Quando um programa é executado, apenas parte de seu código e dados estão na memória principal. O conjunto de frames que um processo possui na memória principal é denominado *working set*. As páginas pertencentes ao processo são mantidas em uma estrutura chamada lista do working set, organizada no esquema FIFO. O tamanho do working set varia conforme a taxa de paginação do processo e da disponibilidade de frames na memória principal. Os tamanhos mínimo e máximo do working set são definidos na criação do processo, mas podem ser alterados durante a sua existência. É importante ressaltar que o conceito de working set no MS Windows não é idêntico ao apresentado no Cap. 10 — Gerência de Memória Virtual.

O MS Windows implementa o esquema FIFO com buffer de páginas como política de substituição de páginas. De forma simplificada, o sistema possui duas listas que funcionam como buffers: a lista de páginas livres (*free page list*) e a lista de páginas modificadas (*modified page list*). As listas de páginas têm a função de reduzir a taxa de page faults dos processos e, principalmente, o número de operações de leitura e gravação em disco.

A lista de páginas livres (LPL) agrupa todos os frames disponíveis para uso na memória principal. Quando um processo necessita de uma nova página e seu working set não atingiu o seu limite máximo, o sistema retira a primeira página da LPL e adiciona ao final da lista do working set do processo (Fig. 14.8b). Por outro lado, quando o working set do processo alcança seu limite máximo, o processo deve ceder a página mais antiga no seu working set para a LPL, antes de receber uma nova página (Fig. 14.8c). Como pode ser observado, a política de substituição de páginas é local, afetando apenas o processo que gerou o page fault.

O destino da página selecionada para substituição vai depender se a página sofre modificação. As páginas não modificadas não precisam de tratamento especial, pois podem ser recuperadas do arquivo executável quando necessário (Fig. 14.8a). Já uma página modificada não pode ser simplesmente descartada quando sai do working set do processo, pois contém dados necessários à continuidade da execução do programa. Nesse caso, a página selecionada não é colocada no final da LPL, mas na lista de páginas modificadas (LPM). A LPM agrupa páginas de todos os processos no sistema e tem a função de adiar a gravação em disco das páginas modificadas (Fig. 14.8d).

A lista de páginas livres, eventualmente, pode ficar com poucas páginas disponíveis para atender à demanda dos processos. Nesse caso, as páginas que estão na LPM são gravadas em disco no arquivo de paginação (Fig. 14.8e) e transferidas para a LPL para aumentar o número de frames livres (Fig. 14.8f). O arquivo de paginação é uma área em disco comum a todos os processos, onde as páginas modificadas são salvas para quando forem posteriormente referenciadas (Fig. 14.8g). As páginas modificadas são gravadas no disco em conjunto e, portanto, apenas uma operação de gravação é efetuada, tornando o processo mais eficiente. O Windows permite múltiplos arquivos de paginação que podem ser distribuídos por vários discos, permitindo um desempenho ainda melhor das operações de leitura e gravação.

A LPL, além de agrupar os frames disponíveis da memória principal, permite também reduzir o impacto do algoritmo FIFO utilizado na política de substituição de páginas. Quando o sistema libera uma página do working set de um processo para a LPL, esta página não é imediatamente utilizada, pois o frame deverá percorrer a lista até chegar ao seu início. Desta forma, as páginas que saem do working set permanecem na memória por algum tempo, permitindo que essas mesmas páginas possam retornar ao working set do processo. Nesse caso, na ocorrência de um page fault o sistema poderá encontrar o frame requerido na LPL, gerando um page fault sem a necessidade de uma operação de leitura em disco (Fig. 14.8h). Por outro lado, quando o frame não é encontrado na LPL, o sistema é obrigado a realizar uma operação de leitura em disco para obter novamente a página referenciada (Fig. 14.8a). De forma semelhante, quando uma página modificada é referenciada o frame pode estar na LPM, dispensando uma operação de leitura em disco (Fig. 14.8i). Caso contrário, o frame deverá ser lido do arquivo de paginação em disco (Fig. 14.8g).

Periodicamente, o sistema verifica o tamanho da LPL e, caso o número de páginas esteja abaixo de um limite mínimo, o tamanho do working set dos processos é reduzido de forma a liberar páginas para a LPL (*working set trimming*). Inicialmente, o sistema seleciona os processos com grandes working sets e que estão inativos por mais tempo ou que apresentem baixas taxas de paginação.

Além da LPL e LPM, o sistema mantém outras três listas para a gerência de memória: lista de páginas ruins, lista de páginas zeradas e lista de páginas em espera. Para controlar todas as páginas e listas na memória principal, a gerência de memória mantém uma base de dados dos frames na memória (*page frame database*), com informações sobre todas as páginas livres e utilizadas.

14.7 Sistema de Arquivos

O MS Windows suporta quatro tipos de sistemas de arquivos: CDFS, UDF, FAT e NTFS. Cada sistema determina como os arquivos e diretórios são organizados, o formato dos nomes dos arquivos, desempenho e segurança de acesso aos dados. O CDFS (CD-ROM File System) oferece suporte a dispositivos como CD-ROMs e DVDs. O UDF (Universal Disk Format) é uma evolução do CDFS, e também é voltado para CDs e DVDs.

O sistema de arquivos FAT (*File Allocation Table*) foi desenvolvido para o sistema MS-DOS e, posteriormente, utilizado nas várias versões do MS Windows. O FAT16 utiliza o esquema de listas encadeadas para estruturar o sistema de arquivos, está limitado a partições de no máxino 4 Gb e apresenta baixo desempenho e segurança. O FAT32 possui a maioria das limitações do sistema de arquivo FAT, porém permite partições de até 8 Tb.

O NTFS (NT File System) foi desenvolvido especialmente para as novas versões do MS Windows, e utiliza o esquema de árvore-B para estruturar o sistema de arquivos, oferecendo alto grau de segurança e desempenho, além de inúmeras vantagens quando comparado aos sistemas FAT, como:

- nomes de arquivos com até 255 caracteres, incluindo brancos e letras maiúsculas e minúsculas;
- partições NTFS dispensam o uso de ferramentas de recuperação de erros;
- proteção de arquivos e diretórios por grupos e ACLs;
- criptografia e compressão de arquivos;
- suporte a volumes de até 16 Exabytes e 2^{32}-1 arquivos por volume;
- ferramentas de desfragmentação e gerência de quotas em disco;
- suporte a Unicode;
- suporte a RAID 0, RAID 1 e RAID 5.

O NTFS trabalha com volumes que são partições lógicas de um disco físico. Um volume pode representar todo o espaço de um disco ou apenas parte do disco físico. Além disso, em um mesmo disco podem ser configurados vários volumes com diferentes sistemas de arquivos, como NTFS e FAT. Esse esquema de particionamento permite o boot de diferentes tipos e versões de sistemas operacionais a partir de um único disco físico.

Um disco, quando formatado, é dividido em setores que são agrupados em clusters. O cluster é uma unidade de alocação de espaço em disco e seu tamanho varia em função do volume, ou seja, quanto maior o volume, maior o tamanho do cluster. O tamanho

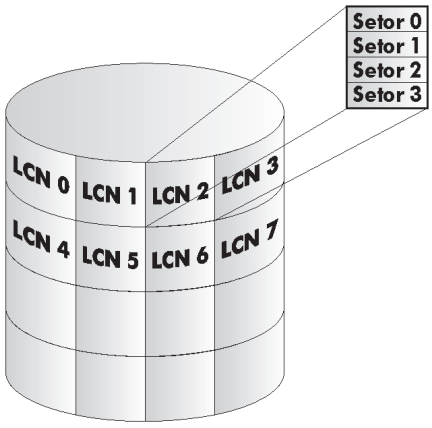

Fig. 14.9 Estrutura lógica do disco.

do cluster influencia na fragmentação interna do volume. Na Fig. 14.9, um volume NTFS pode ser visualizado como uma seqüência de clusters de quatro setores, identificados por um Logical Cluster Number (LCN).

A estrutura de um volume NTFS é implementada a partir de um arquivo chamado Master File Table (MFT), formado por diversos registros de 1 Kb. A Fig. 14.10 apresenta a estrutura do MFT, onde os registros de 0 a 15 são utilizados para mapear os arquivos de controle do sistema de arquivos (arquivos de metadados) e os demais registros são utilizados para mapear arquivos e diretórios de usuários. O registro 0 do arquivo mapeia o próprio MFT.

Os registros no MFT, apesar de possuírem o mesmo tamanho, apresentam formatos diferentes. Um registro tem um header, que identifica o registro, seguido por um ou mais atributos. Cada atributo é formado por um par de cabeçalho e valor, sendo que

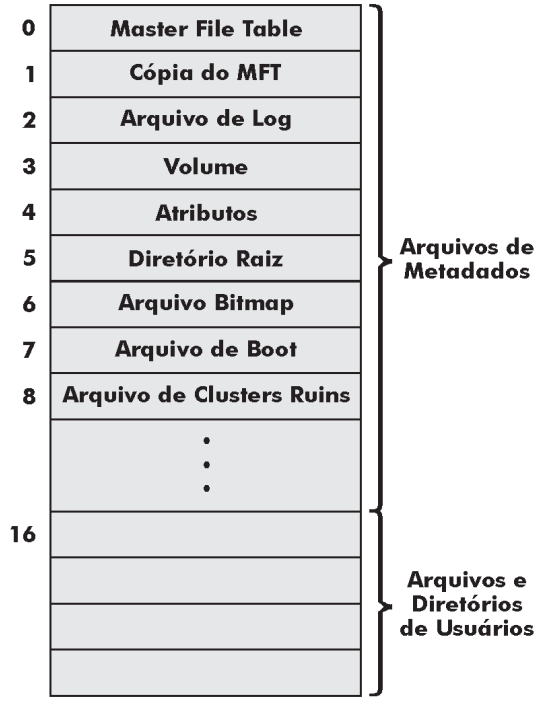

Fig. 14.10 Master File Table.

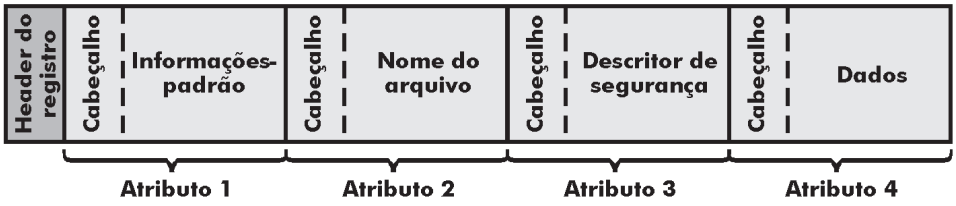

Fig. 14.11 Exemplo de registro para um pequeno arquivo.

o cabeçalho identifica o que o valor representa. O NTFS define 13 tipos diferentes de atributos que podem aparecer em um registro, como nome do arquivo, descritor de segurança e dados do arquivo. No caso de arquivos pequenos, todos os seus atributos, inclusive os dados, podem estar presentes no próprio registro no MFT (Fig. 14.11). No entanto, para a maioria dos arquivos seus atributos podem ultrapassar facilmente o tamanho do registro, sendo colocados em clusters no disco (*nonresident attributes*).

Um arquivo em disco é formado por uma seqüência de clusters não necessariamente contíguos no disco. Por questões de desempenho, sempre que possível o sistema tentará alocar todos os clusters que compõem o arquivo seqüencialmente no disco. Quando não for possível, o arquivo será formado por vários conjuntos de clusters contíguos, sendo que cada conjunto é chamado de *extent*. O número de extents indica o grau de fragmentação do arquivo. Para mapear os extents de um arquivo, o NTFS registra no MFT a posição inicial do extent no disco, utilizando o LCN, e quantos clusters contíguos compõem o extent. A Fig. 14.12 ilustra o exemplo de um arquivo formado por três extents. Caso o número de extents do arquivo ultrapasse o tamanho do registro no MFT, um ou mais registros adicionais podem ser utilizados.

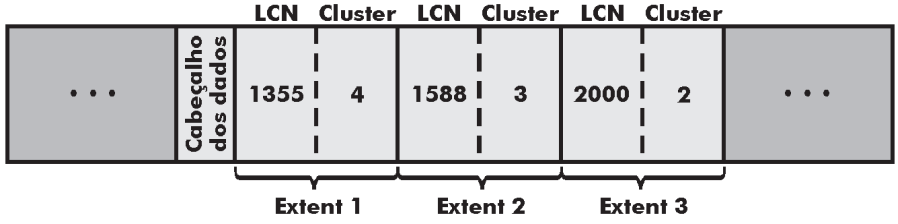

Fig. 14.12 Exemplo de registro de um arquivo.

14.8 Gerência de Entrada e Saída

A gerência de entrada/saída é responsável por receber solicitações de E/S dos diversos threads e repassá-las aos diferentes tipos de dispositivos de E/S, como teclados, impressoras, monitores, discos, CD-ROMs, DVDs e mesmo a rede. O subsistema de E/S foi projetado de forma que novos dispositivos possam ser facilmente conectados ao sistema. Para isso, a gerência de E/S é estruturada em camadas e interage com outros subsistemas, estando intimamente ligada ao gerente de plug-and-play, de energia e de cache, além do sistema de arquivos (Fig. 14.13).

O MS Windows oferece diversas APIs relacionadas à gerência de E/S, sendo a maior parte ligada ao subsistema gráfico, chamado Graphics Device Interface (GDI). O GDI é um conjunto de rotinas que permite a uma aplicação manipular dispositivos gráficos,

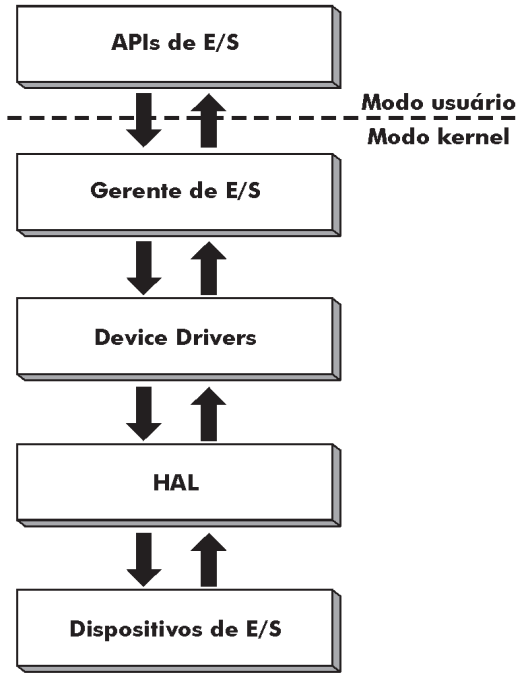

Fig. 14.13 Gerência de E/S.

como monitores e impressoras, independentemente do tipo do dispositivo físico, funcionando como uma interface entre a aplicação e os drivers dos dispositivos. O GDI oferece funções de gerenciamento de janelas, menus, caixas de diálogo, cores, desenhos, textos, bitmaps, ícones e clipboard.

O sistema oferece suporte a operações de E/S assíncronas, ou seja, um thread pode solicitar uma operação de E/S, continuar sua execução e depois tratar o término da solicitação. O término da operação pode ser sinalizada ao thread utilizando um dos diversos mecanismos de sincronização disponíveis no sistema.

O gerente de E/S (I/O Manager) é responsável por receber os pedidos de operações de E/S e repassá-los aos device drivers. Cada solicitação de E/S é representada por uma estrutura de dados chamada I/O Request Packet (IRP), que permite o controle de como a operação de E/S será processada. Quando uma operação de E/S é solicitada, o gerente de E/S cria o IRP e passa a estrutura para o device driver. Terminada a operação, o driver devolve o IRP para o gerente de E/S para completar a operação ou repassá-lo para um outro driver continuar o processamento.

Os device drivers no MS Windows são desenvolvidos a partir de um padrão chamado Windows Driver Model (WDM). O WDM define diversas características e funções que um driver deve oferecer para ser homologado pela Microsoft, como suporte a plug-and-play e a múltiplos processadores, gerência de energia e interface com os objetos do sistema operacional.

O MS Windows trabalha com diferentes tipos de drivers para implementar diversas funções, como a emulação de aplicações MS-DOS (virtual device drivers), interface com o subsistema gráfico GDI (display e printer drivers), implementação dos sistemas de arquivos (file system drivers), funções de filtragem (filter drivers) e controle de dispositivos de E/S (hardware device drivers).

Os device drivers podem ter acesso diretamente ao dispositivo de E/S, como em um driver de disco, ou podem trabalhar em camadas, separando funções ou implementando funções complementares. Por exemplo, as operações de compressão, criptografia e tolerância a falhas de discos são implementadas através de drivers específicos (filter driver) acima dos drivers ligados aos dispositivos de E/S. Os sistemas de arquivos NTFS e FAT também são implementados através de drivers especiais que recebem solicitações do gerente de E/S e as repassam para os drivers de disco correspondentes.