Sistemas operacionais devem também levar em conta o fato de que computadores são geralmente conectados a redes. Hoje em dia, com a World Wide Web, assumimos comunicação em rede como certa. Redes são discutidas em detalhes em um capítulo mais adiante, mas devemos reconhecer aqui o efeito que comunicação em rede tem sobre sistemas operacionais. Tal comunicação é outro recurso que um SO deve suportar.

Um sistema operacional é responsável por comunicação com uma variedade de dispositivos. Usualmente, essa comunicação é alcançada com a ajuda de um controlador* de dispositivo, um pequeno programa que "sabe" o modo que um dispositivo particular espera receber e enviar informação. Com controladores de dispositivos, cada sistema operacional não precisa mais saber sobre todo dispositivo com o qual seria possível esperar que ele viesse a se comunicar no futuro. É outro belo exemplo de abstração. Um controlador apropriado de dispositivo normalmente vem com um novo hardware e a maioria dos controladores mais atualizados pode ser obtida normalmente de forma gratuita a partir do sítio *Web* do fabricante.

Um último aspecto de sistemas operacionais é a necessidade de suportar sistemas de tempo real. Um **sistema de tempo real** é aquele que deve fornecer um **tempo de resposta** mínimo garantido ao usuário. Isto é, o tempo entre receber um estímulo e produzir uma resposta deve ser cuidadosamente controlado. Respostas em tempo real são cruciais em softwares que, por exemplo, controlem um robô, um reator nuclear ou um míssil. Embora todos os sistemas operacionais reconheçam a importância de tempo de resposta, um sistema operacional de tempo real se esforça para otimizá-lo.

Sistema de tempo real Um sistema no qual tempo de resposta é crucial, dada a natureza do domínio de aplicação

Tempo de resposta O tempo decorrido entre receber um estímulo e produzir uma resposta

10.2 Gerenciamento de Memória

Vamos rever o que dissemos sobre memória principal no Capítulo 5. Todos os programas são armazenados em memória principal quando são executados. Todo dado referenciado por estes programas também é armazenado em memória principal, de modo que ele possa ser acessado. A memória principal pode ser vista como uma área grande e contígua de espaço dividida em grupos de 8, 16 ou 32 bits. Cada byte ou palavra de memória tem um endereço correspondente, que é simplesmente um inteiro que identifica unicamente aquela parte particular de memória. Veja a Figura 10.3. O primeiro endereço de memória é 0.

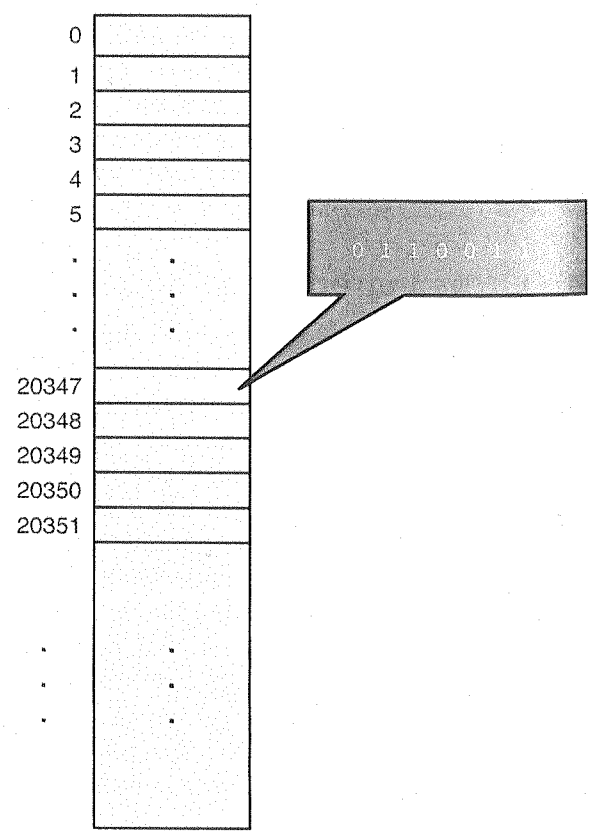

FIGURA 10.3 Memória é um conjunto contíguo de bits referenciados por endereços específicos

Driver no original. (N.T)

Antes, neste capítulo, afirmamos que em um ambiente de multiprogramação, múltiplos programas (e seus dados) são armazenados em memória principal ao mesmo tempo. Assim, sistemas operacionais devem empregar técnicas para realizar as seguintes tarefas:

- Registrar onde e como um programa reside em memória.
- Converter endereços lógicos de programas em endereços reais de memória.

Um programa tem referências a variáveis e a outras partes do código do programa. Quando o programa é compilado, essas referências são convertidas nos endereços de memória onde dados e código residem. Mas, já que não sabemos onde exatamente um programa será carregado em memória principal, como podemos saber qual endereço usar para alguma coisa?

A solução é usar dois tipos de endereços: endereços lógicos e endereços físicos. Um **endereço lógico** (algumas vezes chamado de endereço virtual ou relativo) é um valor que especifica uma localização genérica, relativa ao programa, mas não à realidade de memória principal. Um **endereço físico** é um endereço real no dispositivo de memória principal, como mostrado na Figura 10.3.

Endereço lógico Uma referência a um valor armazenado, relativa ao programa que faz a referência

Endereço físico Um endereço real no dispositivo de memória principal

Quando um programa é compilado, uma referência a um identificador (tal como um nome de variável) é alterada para um endereço lógico. Quando, afinal, o programa é carregado em memória, cada endereço lógico é traduzido para um endereço físico específico. O mapeamento de um endereço lógico em um endereço físico é chamado de **ligação de endereço**. Quanto mais esperarmos para ligar um endereço lógico a um endereço físico, mais flexibilidade teremos. Endereços lógicos permitem que um programa seja movido pela memória ou carregado em diferentes locais, em diferentes momentos. Desde que mantenhamos registro de onde o programa está armazenado, podemos sempre determinar o endereço físico que corresponde a qualquer endereço lógico dado. Para simplificar nossos exemplos neste capítulo, efetuaremos cálculos de ligação de endereços em base 10.

Ligação de endereço O mapeamento de um endereço lógico em um endereço físico

As seções a seguir examinam os princípios subjacentes a três técnicas:

- Gerenciamento de memória contígua única
- Gerenciamento de memória particionada
- Gerenciamento de memória paginada

Gerenciamento de Memória Contígua Única

Vamos inicialmente manter as coisas simples assumindo que existam apenas dois programas em memória: o sistema operacional e o programa aplicativo que queremos executar. Dividiremos a memória principal em duas seções, uma para cada programa, como mostrado na Figura 10.4. O sistema operacional obtém o espaço que ele precisar e ao programa aplicativo é alocado o restante.

Essa abordagem é chamada de **gerenciamento de memória contígua única** porque o programa aplicativo inteiro é carregado em uma grande área de memória. Apenas um programa, além do sistema operacional, pode ser processado por vez. Para ligar endereços, tudo o que temos que considerar é a localização do sistema operacional.

Gerenciamento de memória contígua única A abordagem de gerenciamento de memória na qual um programa é carregado em uma área contígua de memória

Neste esquema de gerenciamento de memória, um endereço lógico é simplesmente um valor inteiro relativo ao ponto inicial do programa. Isto é, endereços lógicos são criados como se o programa fosse carregado na localização 0 da memória principal. Logo, para produzir um endereço físico, somamos um endereço lógico ao endereço de início do programa na memória principal física.

FIGURA 10.4 Memória principal dividida em duas seções

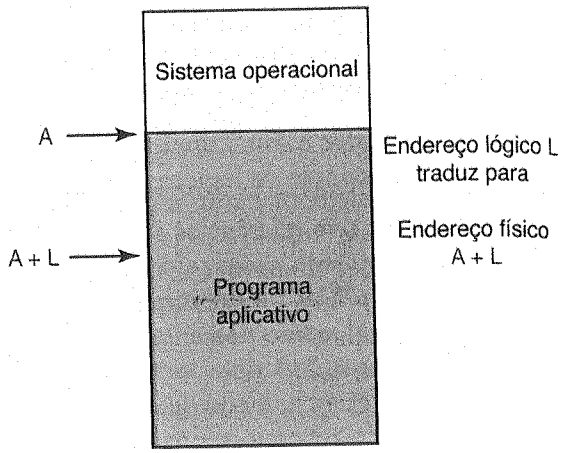

FIGURA 10.5 Ligando um endereço lógico a um endereço físico

Sejamos um pouco mais específicos: se o programa for carregado começando no endereço A, então o endereço físico correspondente ao endereço lógico L será A + L. Veja a Figura 10.5. Vamos incluir números para tornar este exemplo mais claro. Suponha que o programa seja carregado em memória começando no endereço 555555. Quando um programa usa o endereço relativo 222222, sabemos que ele na verdade estará se referindo ao endereço 777777 na memória principal física.

Na verdade, não importa que endereço L é. Desde que mantenhamos o registro de A (o endereço inicial do programa), sempre poderemos traduzir um endereço lógico em um endereço físico.

Neste momento, você pode estar dizendo: "se trocássemos a posição do sistema operacional com a do programa, então os endereços lógico e físico para o programa seriam os mesmos". Isto é verdade. Mas então você teria outras coisas com que se preocupar. Por exemplo, um esquema de gerenciamento de memória deve sempre levar em conta segurança. Em particular, em um ambiente de multiprogramação, devemos evitar que um programa acesse qualquer endereço além do espaço de memória a ele alocado. Com o sistema operacional alocado na posição 0, todos os endereços lógicos para o programa são válidos, a menos que eles excedam o próprio limite da memória principal. Se movermos o sistema operacional para baixo do programa, deveremos assegurar que um endereço lógico não tente acessar o espaço de memória dedicado ao sistema operacional. Isto não é difícil, mas adiciona alguma complexidade de processamento.

A vantagem da abordagem de gerenciamento de memória contígua única é que ela é simples de implementar e gerenciar. No entanto, é quase certo que ela desperdice tanto espaço de memória como tempo de CPU. É pouco provável que um programa aplicativo necessite de todo o espaço de memória não usado pelo sistema operacional e tempo de CPU é desperdiçado quando o programa tem que esperar por algum recurso.

Apple acompanha arquivos livres de DRM
Em janeiro de 2009, a Apple anunciou que, após seis anos, a Loja iTunes® logo deixaria por completo de vender música incluindo restrições DRM (*Digital Rights Management* – Gerenciamento de Direitos Digitais). Por bastante tempo a Apple vendeu faixas selecionadas livres de DRM como faixas iTunes Plus. Essas faixas originalmente incluíam músicas de apenas um dos principais selos de gravadoras, EMI. A nova biblioteca livre de DRM da Apple agora inclui faixas da Sony BMG, Warner Music e Universal. Embora a Amazon® venda há muito tempo faixas livres de DRM de todos os quatro selos, os arquivos de áudio da Amazon são no formato MP3, enquanto os arquivos da Apple usam AAC (*Advanced Audio Coding* – Codificação Avançada de Áudio). Arquivos AAC geralmente têm melhor qualidade sonora do que arquivos MP3. No final de março de 2009, a Apple já tinha tornado todas as suas faixas disponíveis como arquivos livres de DRM.

Gerenciamento de Memória Particionada

Técnica de partição fixa A técnica de gerenciamento de memória na qual a memória é dividida em um número específico de partições nas quais programas são carregados

Técnica de partição dinâmica A técnica de gerenciamento de memória na qual a memória é dividida em partições conforme for necessário acomodar programas

Uma abordagem mais sofisticada é ter mais de um programa aplicativo em memória ao mesmo tempo, compartilhando espaço em memória e tempo de CPU. Nesse caso, a memória deve ser dividida em mais de duas partições. Duas estratégias podem ser usadas para particionar memória: partições fixas e partições dinâmicas. Com a **técnica de partição fixa**, a memória principal é dividida em um número específico de partições. As partições não precisam ter o mesmo tamanho, mas os tamanhos delas são fixados quando o sistema operacional é inicializado. O SO mantém uma tabela de endereços na qual cada partição começa e o tamanho da partição.

Com a **técnica de partição dinâmica**, as partições são criadas para se ajustar às necessidades únicas dos programas. A memória principal é vista inicialmente como uma grande partição vazia. À medida que programas são carregados, o espaço é "fatiado", usando apenas o espaço necessário para acomodar o programa e deixando um nova partição menor e vazia, que poderá ser usada por outro programa, posteriormente. O sistema operacional mantém uma tabela de informação de partição, mas com partições dinâmicas as informações de endereço mudam à medida que programas vêm e vão.

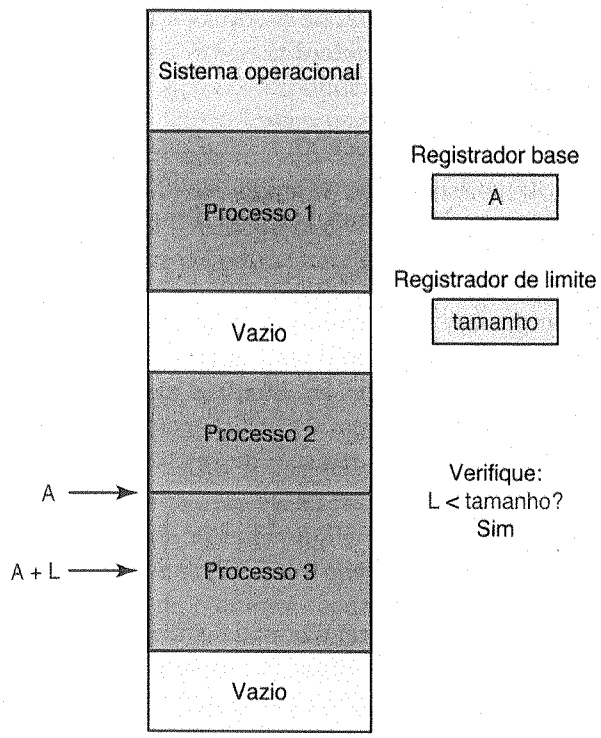

FIGURA 10.6 Resolução de endereços em gerenciamento de memória particionada

Em qualquer instante de tempo, tanto em partições fixas como dinâmicas, a memória é dividida em um conjunto de partições, algumas vazias e algumas alocadas para programas. Veja a Figura 10.6.

A ligação de endereços é basicamente a mesma para as partições fixas e dinâmicas. Tal como a abordagem de gerenciamento de memória contígua única, um endereço lógico é um inteiro relativo ao ponto de partida 0. Existem vários meios de um SO lidar com os detalhes da tradução de endereços. Uma forma é usar dois registradores de propósitos especiais da CPU para ajudar a gerenciar endereçamento. Quando um programa se torna ativo na CPU, o SO armazena o endereço do início da partição daquele programa no **registrador base**. De forma similar, o tamanho da partição é armazenado no **registrador de limites**. Quando um endereço lógico é referenciado, ele é primeiramente comparado ao valor contido no registrador de limites para assegurar que ele esteja no espaço de memória alocado àquele programa. Se estiver, o valor do endereço lógico será somado ao valor no registrador base para produzir o endereço físico.

Registrador base Um registrador que mantém o endereço de início da partição corrente

Registrador de limites Um registrador que mantém o tamanho da partição corrente

Qual partição devemos alocar a um novo programa? Há três abordagens gerais para seleção de partição:

- *Primeira escolha*, na qual o programa é alocado na primeira partição grande o suficiente para acomodá-lo
- *Melhor escolha*, na qual o programa é alocado na menor partição com tamanho suficiente para acomodá-lo
- *Pior escolha*, na qual o programa é alocado na maior partição com tamanho suficiente para acomodá-lo

Não faz sentido usar a pior escolha em partições fixas, pois ela desperdiçaria as partições maiores. A primeira escolha e a melhor escolha funcionam para partições fixas. Com partições dinâmicas, contudo, a pior escolha frequentemente funciona melhor, já que ela deixa a maior partição possível vazia, que poderá acomodar outro programa mais tarde.

Quando um programa termina, a tabela de partição é atualizada para refletir o fato de que a partição agora está vazia e disponível para um novo programa. Com partições dinâmicas, partições vazias consecutivas são fundidas em uma grande partição vazia.

Gerenciamento de memória particionada torna eficiente o uso de memória principal mantendo diversos programas em memória de uma vez. Mas tenha em mente que um programa deve caber inteiramente em uma partição. Partições fixas são mais fáceis de gerenciar que aquelas dinâmicas, mas restringem as oportunidades disponíveis para novos programas. O sistema pode ter memória disponível o suficiente para acomodar o programa, mas não em uma única partição livre. Com