Unidade II

5 GERENCIAMENTO DE MEMÓRIA

Diferentemente da realidade atual dos sistemas computacionais, a memória principal era tratada como um recurso escasso e caro. Dessa forma, os projetistas desenvolviam o sistema operacional para que não utilizasse muito espaço na memória e realizasse um uso otimizado dos recursos do computador.

Para sistemas monoprogramáveis, a gerência da memória é implementada de forma bastante simplificada. Por sua vez, nos sistemas multiprogramáveis, ela se torna crítica, devido à necessidade de se maximizar a quantidade de usuários e processos utilizando o espaço da memória principal de forma otimizada.

Nesta unidade, serão exibidos os esquemas básicos de gerência da memória principal, apresentando suas vantagens, desvantagens e como evoluíram historicamente. Adicionalmente, introduziremos o mecanismo de gerência de memória virtual nos sistemas operacionais modernos.

Mesmo atualmente, com a redução de custo e consequente aumento da capacidade da memória principal, seu gerenciamento é um dos fatores mais importantes no projeto de sistemas operacionais.

Idealmente, os programadores desejam que a memória de um sistema computacional seja de alta capacidade, de baixo custo, de altíssima velocidade de acesso e com baixo tempo de resposta e não volátil, isto é, que não se apague na ausência de energia elétrica. Apesar da evolução da memória, a tecnologia atual não comporta a construção de tais memórias.

5.1 Tipos de memória

Em um sistema computacional, existem diversos tipos de memória com características e particularidades próprias, que variam muito em função do tamanho e do tempo de acesso à memória. O objetivo de todas as memórias é igual: armazenar informação.

Os locais de armazenamento internos dos dados em um computador são os registradores do processador, a memória cache tanto interna ao processador, chamada de cache L_1, como o cache externo da placa mãe, conhecido como L_2, e a memória principal ou memória de acesso aleatório, do inglês *random access memory* (RAM). Adicionalmente, existem discos e unidades de armazenamento externos, tais como pen drives, CD-ROMs, DVD-ROMs e fitas magnéticas que possuem a função de armazenar informação de forma não volátil.

Esses dispositivos de hardware são construídos utilizando diversas tecnologias e possuem características distintas, por exemplo, a capacidade de armazenamento, a velocidade de operação, o custo por tamanho de memória armazenado, o consumo energético e se a memória é volátil ou não volátil. A figura a seguir apresenta a hierarquia de memórias, normalmente representada como uma pirâmide.

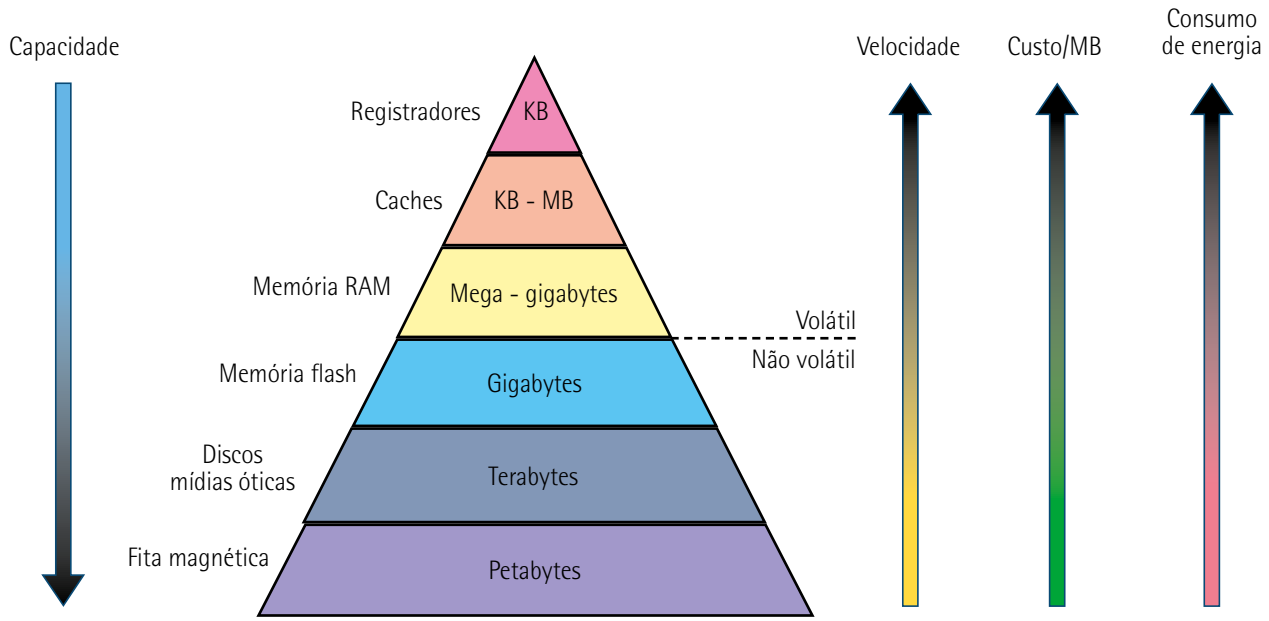

Figura 45 – Hierarquia de memórias

Fonte: Maziero (2019, p. 162).

Em relação à velocidade de uma memória, existem duas medidas mais utilizadas, que são o tempo de acesso ou latência e a taxa de transferência. Entende-se tempo de acesso como o período necessário para começar uma transferência de dados indo ou vindo de um meio de armazenamento. Por sua vez, a taxa de transferência representa a quantidade de bytes transmitidos ou recebidos por segundo que são lidos ou escritos por um tipo de memória após o início da transferência de dados.

A memória principal do computador inclui uma área de RAM composta de uma grande sequência de bytes, uma unidade de memória utilizada pelo processador. Cada byte da memória RAM possui um endereço, que é usado para acessá-lo, e o tempo de acesso é praticamente o mesmo para todos os endereços de memória. Um computador moderno padrão possui alguns gibabyte (Gb) de memória RAM, isto é, mais de um bilhão de posições de memória diferentes. Essa memória é utilizada para manter o sistema operacional e os processos em execução, bem como algumas áreas de finalidades específicas, por exemplo, buffers de dispositivos de E/S. A quantidade de memória RAM disponível em um sistema computacional forma o espaço de memória física.

Lembrete

O processador obtém acesso à memória principal através de barramentos de dados, de endereços e de controle. O barramento de endereços possui um número fixo de vias, que define a quantidade total de endereços de memória que podem ser gerados pelo processador, e um barramento de dados com n vias consegue gerar 2n endereços. O conjunto de endereços de memória que um processador pode representar é denominado de **espaço de endereçamento**.

5.2 Funções do gerenciamento de memória

Uma das tarefas do gerenciador de memória é organizar a hierarquia de memória, de forma a identificar os espaços livres e ocupados, e alocar ou localizar tanto processos quanto dados nela. Outra função sua é controlar as partes que estão em uso e as que estão livres para alocação. Com isso, o gerenciamento de memória pode alocar memória aos processos, quando necessário, e liberar memória no encerramento de um processo, atualizando o status da memória.

De acordo com Machado e Maia (2013), o papel da gerência de memória é tentar manter na memória principal a maior quantidade de processos alocados e assim permitir a maximização do compartilhamento do processador e demais recursos. Mesmo quando não exista mais espaço livre na memória principal, o sistema deve permitir que novos processos sejam aceitos e executados. Para isso ser possível, o sistema realiza transferência temporária de processos residentes da memória principal para a memória secundária, ou seja, para o disco rígido. Portanto, há a liberação de espaço para novos processos e tal mecanismo é denominado swapping.

Em ambientes de multiprogramação, é necessário que o sistema operacional proteja as áreas de memória já utilizadas por cada processo residente e a área de memória onde está alocado o próprio sistema. Quando um programa realizar uma tentativa de acesso indevido à memória, o sistema operacional deverá impedir o acesso não autorizado de todas as formas.

5.2.1 Swapping

Em algumas situações, não há espaço suficiente na memória principal para conter todos os processos correntemente ativos. A técnica de swapping foi criada para resolver esse problema, fazendo com que os processos excedentes sejam mantidos no disco e trazidos para execução dinamicamente. Assim, há troca de memórias entre os programas que são da memória principal e armazenados temporariamente no disco rígido e, posteriormente e na existência de espaço livre, saem do disco rígido para a memória principal.

A figura 46 representa o funcionamento do swapping. O sistema escolhe um processo residente na memória, o qual será transferido para a memória secundária, normalmente o disco rígido, e esta etapa é denominada de swap out. Quando existe uma partição livre na memória principal, o processo que saiu da memória principal e foi transferido para memória secundária é carregado de volta na memória principal, na etapa denominada swap in. Com isso, o processo poderá continuar a ser executado como se nada tivesse acontecido.

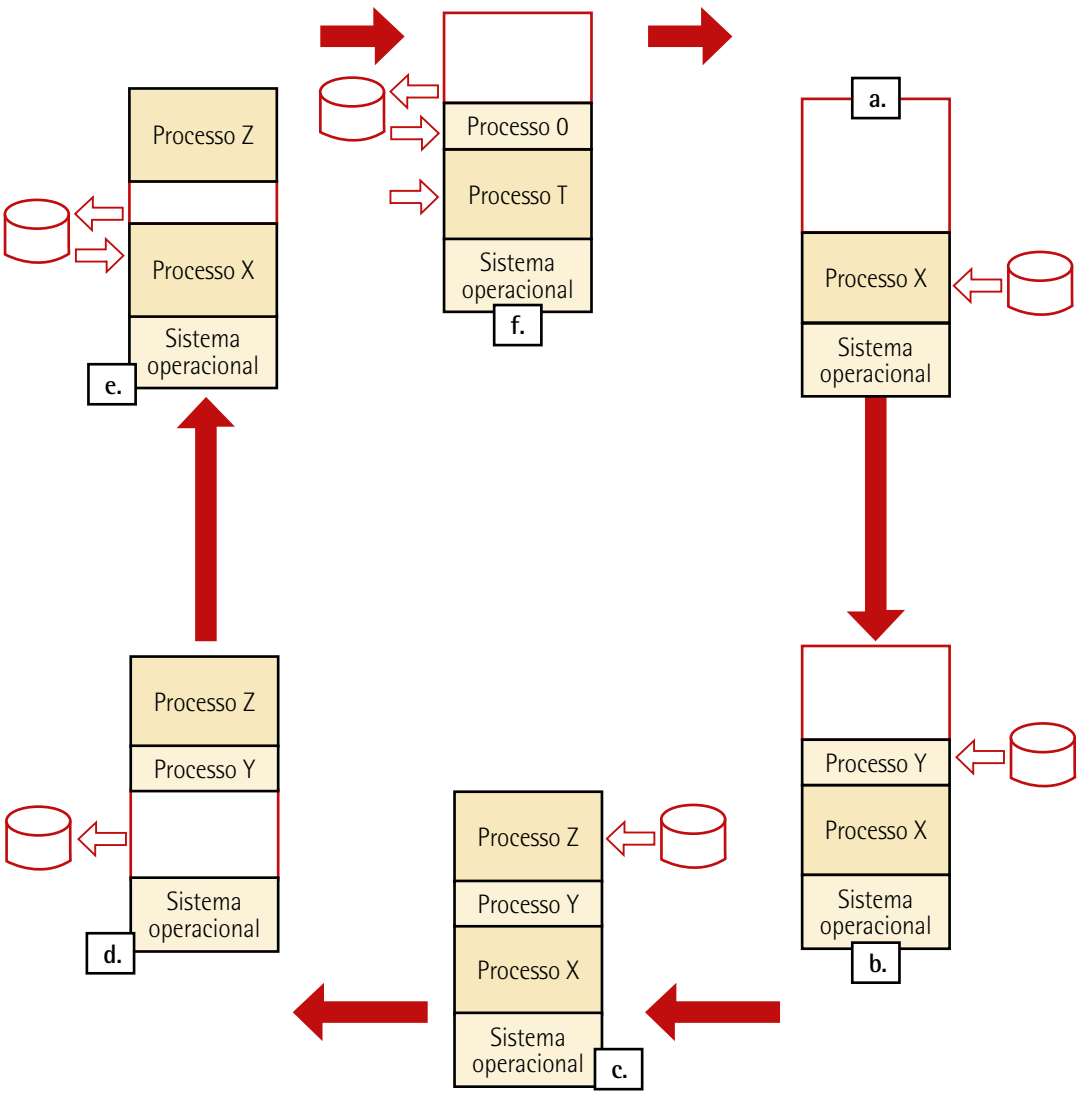

Figura 46 – Ciclo de troca dos processos entre memória principal e secundária

Fonte: Furukawa e Nunes (2011, p. 62).

No item (a) da figura, inicialmente, o sistema operacional ocupando a parte mais baixa da memória (permanecerá nesta posição) e, logo em seguida, temos o processo X ocupando uma parte da memória disponível. Depois, no item (b), um novo processo Y é criado ou trazido do disco duro e posicionado na memória logo acima do processo X. Em (c), um novo processo Z é adicionado. Nesse momento não há mais espaço disponível na memória principal para novos processos. Em (d), o processo X

fica ocioso, então, é enviado para o disco rígido. Em (e)/(f), temos outros processos sendo trocados e o ciclo vai sendo executado até que novos processos entrem e disputem o tempo de CPU e memória e/ou que processos terminados sejam eliminados.

Observação

Através do processo de swapping, quando um processo for maior que a área livre, ou não existir nenhuma área de memória disponível, esse processo será transferido para disco e ficará na memória secundária até que seja liberada memória principal suficiente para o carregamento do programa.

Com a introdução da técnica de swapping, é possível maior compartilhamento da memória principal e, consequentemente, maior uso dos recursos do sistema computacional. Entretanto, essa técnica traz um elevado custo das operações de E/S (swap in/swap out). Quando existe pouca memória disponível, o sistema pode ficar comprometido com a realização de swapping, deixando de executar outras tarefas e impedindo a execução dos demais processos residentes. Trata-se de uma situação considerada como crítica na gerência de memória, que é denominada de thrashing.

Segundo Machado e Maia (2013), o loader ou carregador é o utilitário do sistema operacional com a função de carregar na memória principal um programa a ser executado. O procedimento de carga varia com o código gerado pelo linker e, em função dele, o loader é classificado como do tipo absoluto ou relocável.

No caso de um código executável do tipo absoluto, o loader somente precisa conhecer o endereço de memória inicial e o tamanho do módulo para realizar o carregamento. Desse modo, ele transfere o programa da memória secundária para a memória principal e inicia sua execução (loader absoluto). Com este tipo de código, não é possível realizar a operação de swapping.

No caso do código relocável, o programa pode ser alocado em qualquer posição de memória, e o loader é responsável pela relocação no momento do carregamento (loader relocável).

Para a implementação da técnica de swapping, é necessário que o sistema implemente a relocação dinâmica, pois o programa pode entrar e sair da memória principal e ser alocado em diferentes endereços de memória. Essa operação é feita por meio de um registrador denominado de registrador de relocação. Assim que o programa é colocado na memória, tal registrador obtém o endereço inicial da posição de memória a ser ocupada pelo programa.

Durante a execução do código, quando existe referência a algum endereço, será adicionada essa referência com o valor de endereço inicial e determinado o endereço físico, como mostrado na figura a seguir. Com isso, será possível carregar o programa em qualquer posição de memória.

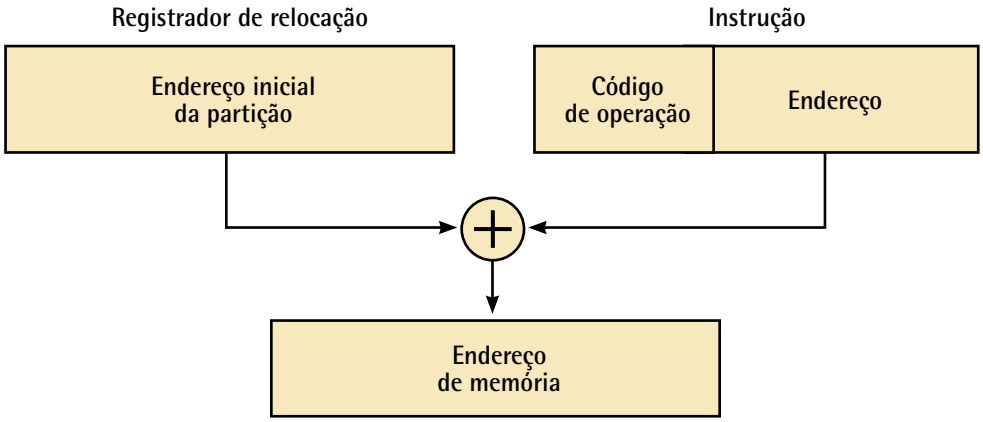

Figura 47 – Relocação dinâmica

Fonte: Machado e Maia (2013, p. 157).

5.2.2 Unidade de gerenciamento de memória (MMU)

O espaço de endereçamento lógico é o conjunto de todos os endereços lógicos gerados por um programa. Analogamente, o espaço de endereço físico é composto de todos os endereços físicos correspondentes a esses endereços lógicos. A unidade de gerenciamento de memória ou *memory management unit* (MMU) é um dispositivo de hardware que transforma endereços virtuais em endereços físicos.

Endereços físicos ou reais são os endereços dos bytes de memória física do computador. Eles são definidos pela quantidade de memória disponível na máquina. Já os **endereços lógicos** ou virtuais são aqueles de memória usados pelos processos e pelo sistema operacional e, portanto, usados pelo processador durante a execução. Tais endereços são definidos de acordo com o espaço de endereçamento do processador.

O espaço de endereço do processo representa o conjunto de endereços lógicos a que um processo faz referência em seu código-fonte. Há três tipos de endereços usados em um programa, antes e depois da memória ser alocada: endereços simbólicos, endereços relativos e endereços físicos. Os endereços simbólicos são aqueles utilizados em um código-fonte, por exemplo, nomes das variáveis, constantes e rótulos de instrução.

Para os endereços relativos, no momento da compilação, um compilador converte endereços simbólicos em endereços relativos. Já para os endereços físicos, o carregador produz esses endereços no momento em que um programa é carregado na memória principal.

Os programas manipulam endereços lógicos e nunca conhecem o endereço físico real. A MMU suporta o sistema operacional na realização do mapeamento dos endereços da memória física e da memória virtual, permitindo, assim, a movimentação eficaz das partes dos programas da memória virtual para o disco ou vice-versa.

A figura a seguir apresenta a localização da MMU como parte do chip da UCP, como é na maior parte dos computadores atuais.

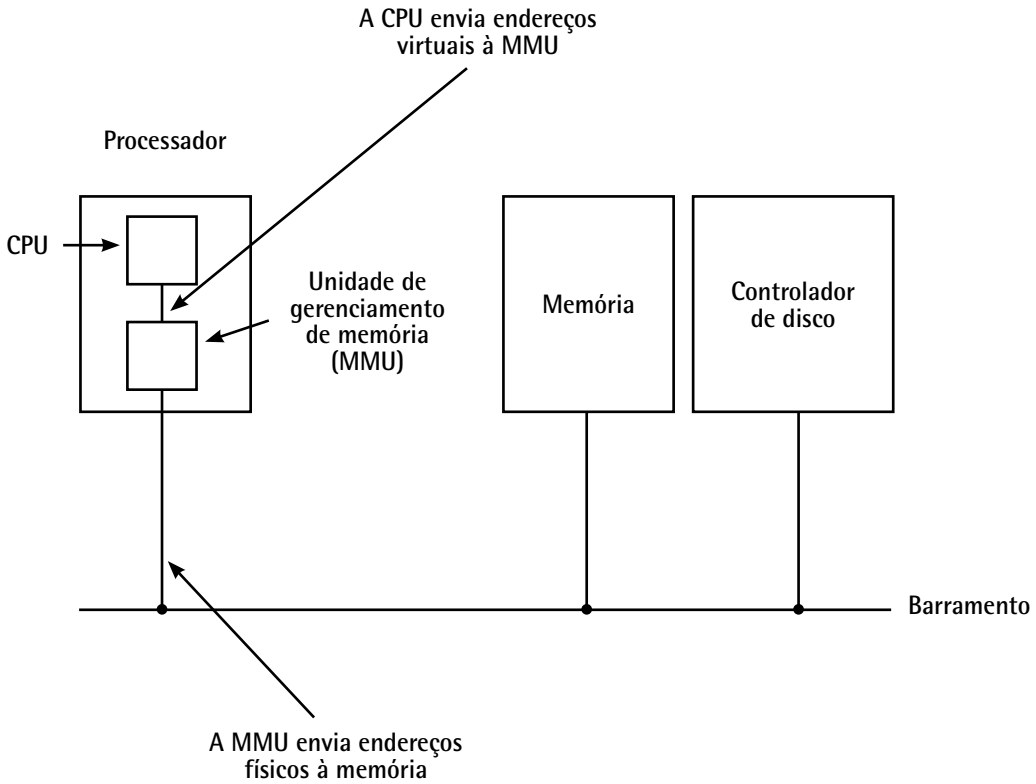

Figura 48 – Posição e função da MMU

Fonte: Tanenbaum e Bos (2016, p. 135).

5.3 Gerenciamento básico de memória

De acordo com Tanenbaum e Woodhull (2008), os sistemas de gerenciamento de memória podem ser divididos em duas classes fundamentais: aqueles que utilizarão exclusivamente a memória principal para alocar os processos e os outros que alternam os processos entre a memória principal e o disco durante a execução. Essa troca de memória é conhecida como swapping.

5.3.1 Sistemas monoprogramáveis

A forma de gerenciamento de memória mais simplificada ocorre quando é executado apenas um programa por vez, compartilhando a memória exclusivamente entre ele e o sistema operacional. Na figura a seguir são apresentadas três variadas formas de organização da memória principal que será dividida para alocar o sistema operacional e o programa do usuário.

O sistema operacional pode estar na parte inferior da memória na RAM; como se vê em (a) da figura a seguir, pode estar em uma memória somente de leitura *read-only memory* (ROM), na parte superior

da memória, como se observa em (b), ou os drivers de dispositivo podem estar na parte superior da memória em uma ROM e o restante do sistema na RAM abaixo dela, como é visto em (c).

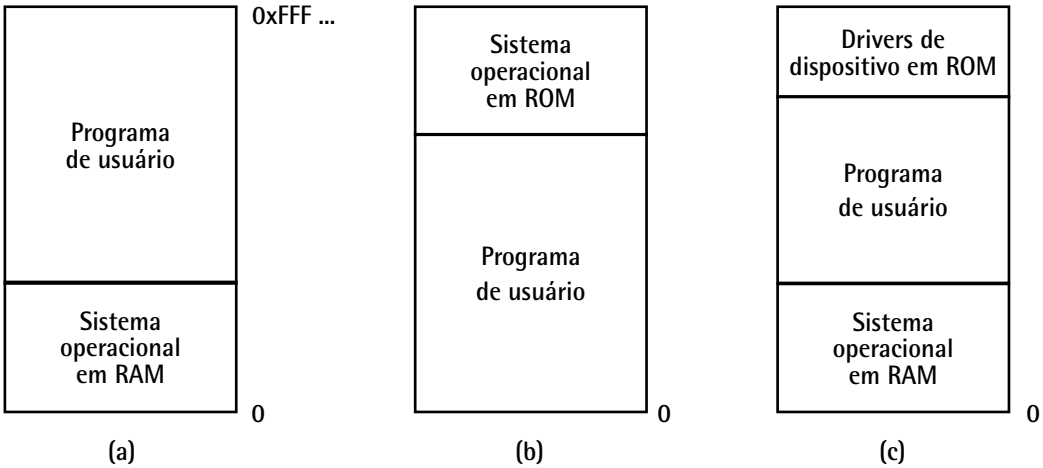

Figura 49 – Formas do gerenciamento de memória para sistemas monoprogramáveis

Fonte: Tanenbaum e Woodhull (2008, p. 347).

O primeiro modelo foi usado em computadores de grande porte e em minicomputadores, entretanto isso não mais ocorre. O segundo modelo foi empregado em pequenos computadores de mão ou palmtops e em sistemas embarcados. Já o terceiro modelo foi aplicado nos primeiros computadores pessoais, por exemplo, no sistema MS-DOS. Nele, parte do sistema que fica na ROM é chamada de sistema básico de entrada e saída (*basic input output system* - Bios).

5.3.2 Sistemas multiprogramáveis com partições fixas

Com a evolução dos programas utilitários do sistema operacional, o código do programa resultante passou a ser relocável, em vez de fazer uma referência de endereço absoluto. No código relocável, as referências a endereços existentes no programa são relativas ao início do código, e não estão atreladas a endereços físicos de memória. Com isso, os programas podem ser executados a partir de qualquer partição.

Uma maneira simplificada do gerenciamento de memória para sistemas multiprogramáveis consiste em simplesmente dividir a memória em até n partições, que não precisam ter o mesmo tamanho. Esse particionamento pode ser feito manualmente, por exemplo, quando o sistema é inicializado, sendo esse tipo de gerência de memória conhecido como alocação particionada estática ou fixa.

Quando ocorre a chegada de um processo, ele pode ser colocado na fila de entrada da menor partição com tamanho suficiente para contê-lo. Como as partições são fixas nesse esquema, todo espaço não utilizado por um job em uma partição é desperdiçado, enquanto esse job é executado. Na figura a seguir, está representado como é o sistema de partições fixas e filas de entrada separadas.

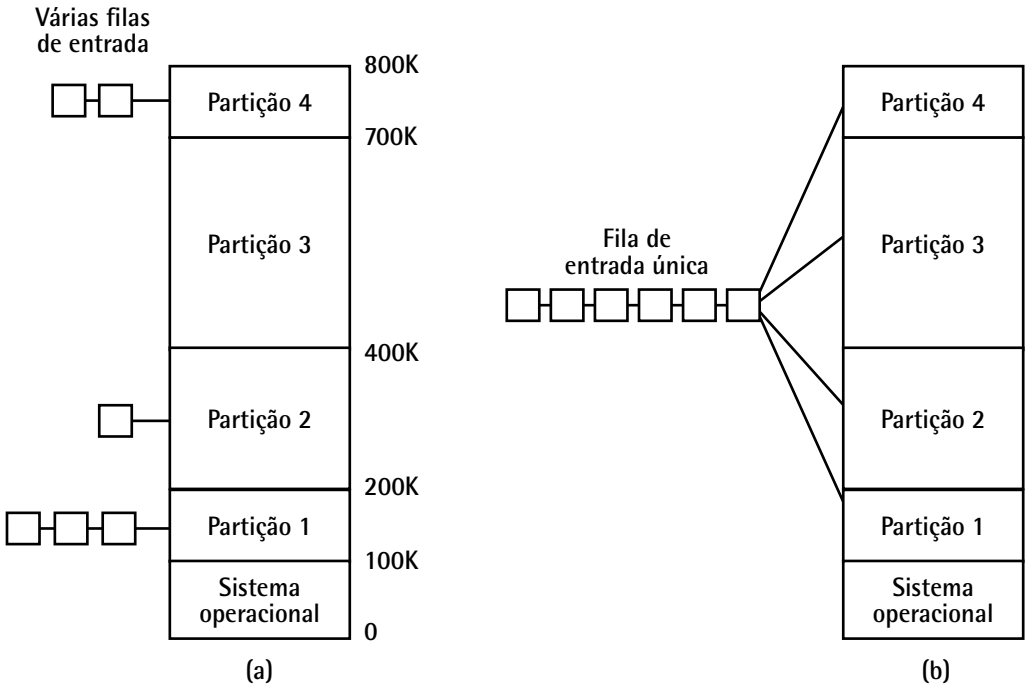

Figura 50 – Multiprogramação com partições fixas de memória (a) com filas de entrada separadas para cada partição, (b) com uma fila única de entrada

Fonte: Tanenbaum e Woodhull (2008, p. 347).

Uma desvantagem existente na ordenação das tarefas recebidas em filas separadas que se torna evidente quando a fila de uma partição grande está vazia, mas a de uma partição pequena está cheia, acontece nas partições 1 e 3 em (a) da figura anterior. Nessa situação, os jobs pequenos são obrigados a esperar para serem alocados na memória, mesmo com partições de memória livre.

Em caso de fila única, como em (b) da figura anterior, assim que uma partição fica desocupada, o job mais próximo do começo da fila, e que caiba na partição vazia, pode ser alocado na partição e executado.

Visando manter o controle sobre quais partições estão alocadas, a gerência de memória elabora uma tabela com o endereço inicial de cada partição, o tamanho da participação e se ela está alocada para algum processo ou não, como mostra a figura a seguir.

Partição	Tamanho	Livre
1	2 Kb	Não
2	5 Kb	Sim
3	8 Kb	Não

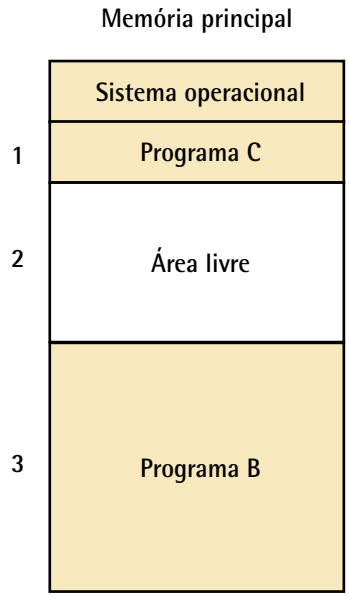

Figura 51– Tabela de alocação de partições

Fonte: Machado e Maia (2013, p. 151).

Quando um programa é selecionado para ser carregado na memória, a tabela de alocação de partições é percorrida pelo sistema operacional, na tentativa de localizar uma partição livre onde o programa possa ser carregado.

Com o advento da multiprogramação, dois problemas básicos são introduzidos no gerenciamento de memória e devem ser resolvidos: realocação e proteção.

Nesse esquema de alocação de memória, existem dois registradores, os quais representam o limite inferior e superior da partição onde o programa está sendo executado. Quando o programa tenta acessar uma posição de memória fora dos limites definidos pelos registradores, ele é paralisado e o sistema operacional envia uma mensagem de violação de acesso.

Quando o tamanho da partição é fixo, dificilmente os programas preenchem as partições onde são colocados e o espaço de memória que sobra será desperdiçado. Esse tipo de problema é denominado de fragmentação interna.

5.3.3 Sistemas multiprogramáveis com partições variáveis

Com o objetivo de reduzir a fragmentação interna, foi eliminado o conceito de partições de tamanho fixo com a introdução da alocação particionada dinâmica ou variável. Nesse esquema, cada programa pode utilizar o espaço de que necessita, tornando essa área sua partição. Dado que os programas usam apenas o espaço necessário, o problema da fragmentação interna não acontece.

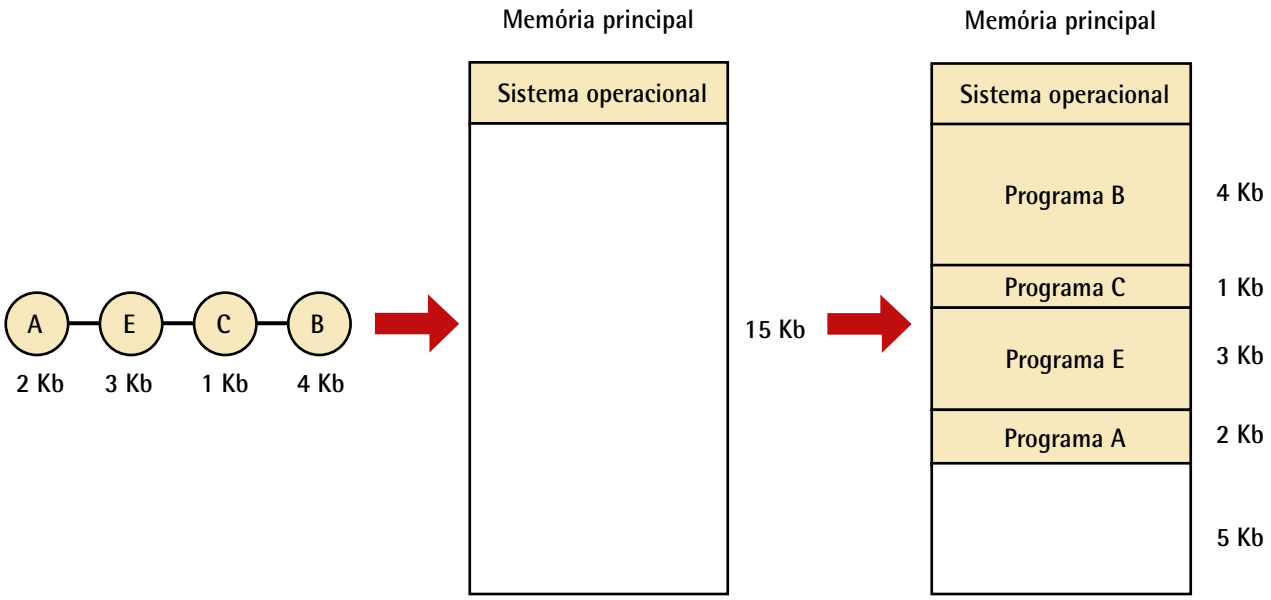

Figura 52 – Alocação particionada dinâmica

Fonte: Machado e Maia (2013, p. 152).

Entretanto, depois de diversas alocações de processos, uma situação que pode acontecer é os espaços de memória se tornarem cada vez menores na memória e os novos programas não conseguirem ser neles alocados. A esse novo problema damos o nome de fragmentação externa.

5.3.4 Estratégias de alocação

Com a intenção de minimizar a ocorrência de fragmentação externa, cada pedido de alocação pode ser analisado para encontrar a área de memória livre que melhor o atenda. A melhor estratégia a ser escolhida por um sistema depende de alguns fatores, e o tamanho dos programas processados é o mais relevante. Independentemente do algoritmo adotado, o sistema possui uma lista de áreas livres, com o endereço e tamanho de cada área.

A estratégia **first-fit** (o primeiro que couber) consiste em escolher a primeira área livre que possua um espaço livre maior ou igual ao solicitado. A vantagem para tal abordagem é a rapidez, sobretudo se a lista de áreas livres for muito longa.

Na estratégia **best-fit** (a que melhor couber), a partição escolhida será aquela na qual resulta o menor espaço sem utilização após a alocação do programa. Para esse algoritmo, o objetivo é minimizar o desperdício de memória, e a listagem de áreas livres está ordenada por tamanho, diminuindo o tempo de busca por uma área desocupada. Como o algoritmo aloca a partição que deixa a menor área livre, existe a tendência de que a memória fique com pequenas áreas não contíguas e, consequentemente, agrave o problema da fragmentação externa.

Por sua vez, a estratégia **worst-fit** (a que pior couber) consiste na escolha da maior área livre possível, de maneira que o excedente não utilizado, seja o suficiente para ser usada em futuras alocações.

Uma variação da estratégia first-fit é a estratégia **next-fit** (o próximo que couber). Nela, percorre-se a lista de áreas a partir da última área alocada ou liberada, para que o uso das áreas livres tenha uma distribuição mais homogênea no espaço de memória.

5.4 Memória virtual

A memória virtual representa uma técnica sofisticada e poderosa de gerência de memória na qual tanto a memória principal quanto a secundária estão agrupadas de modo a oferecer ao usuário a impressão da existir uma memória principal de maior capacidade que a real. Na memória virtual, são utilizados dois tipos de endereços físicos e endereços lógicos.

Um conceito fundamental de memória virtual consiste em não vincular o endereçamento feito pelo programa dos endereços físicos da memória principal. Assim, não existe mais o limite da memória física disponível para programas e suas estruturas de dados, já que eles podem possuir endereços associados à memória secundária.

Adicionalmente, a memória virtual possibilita maior quantidade de processos compartilhando a memória principal, pois apenas partes de cada processo estarão residentes na memória. Outros benefícios dessa técnica são aumentar o grau de multiprogramação e minimizar o problema da fragmentação da memória principal.

Em sistemas que implementam a técnica de memória virtual, o espaço de endereçamento do processo é conhecido como espaço de endereçamento virtual, que identifica o conjunto de endereços virtuais que o processo pode endereçar. De forma análoga, o conjunto de endereços físicos ou reais que o processador pode referenciar é chamado de espaço de endereçamento real.

O mecanismo de tradução do endereço virtual para endereço físico é conhecido como mapeamento. O espaço de endereçamento virtual não possui relação direta com os endereços no espaço real, posto que um programa pode referenciar endereços virtuais que estejam fora dos limites da memória principal. Para que isso ocorra, a memória secundária é utilizada como extensão da memória principal. Na inicialização de um processo, apenas o programa fica parcialmente residente na memória principal e o restante permanece na memória secundária até que seja referenciado.

A figura a seguir ilustra ambos os espaços de endereçamento.

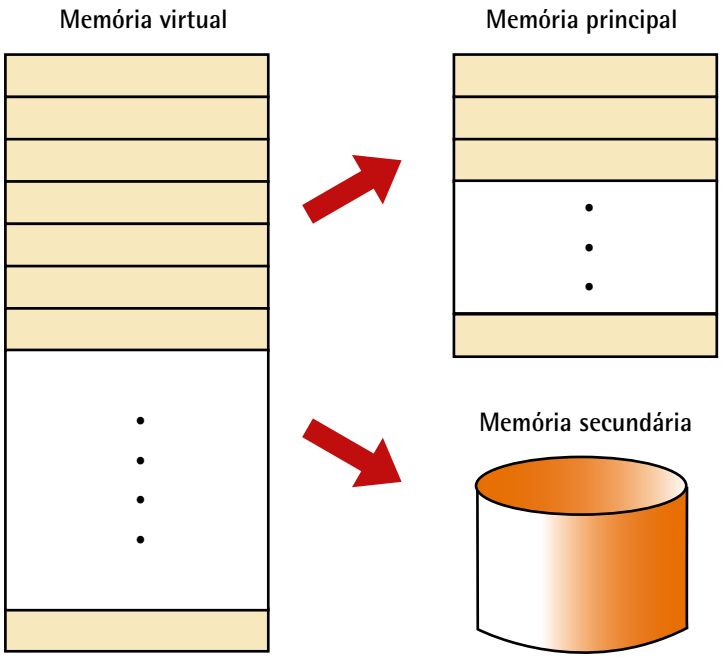

Figura 53 – Espaço de endereçamento virtual

Fonte: Machado e Maia (2013, p. 160).

5.4.1 Memória virtual por paginação

Nesta técnica de gerência de memória, os espaços de endereçamento virtual e real são divididos em blocos de tamanho fixo, que são denominados página. As páginas no espaço real são denominadas de páginas reais ou frames, enquanto as páginas no espaço real são conhecidas como páginas virtuais.

A fim de realizar o mapeamento da memória virtual, são utilizadas as tabelas de páginas. Para cada processo há sua própria tabela de páginas, e cada página virtual do processo gera uma entrada na tabela de páginas (ETP), incorporando informações de mapeamento que possibilitem a localização da página real correspondente.

A figura a seguir apresenta a tabela de páginas.

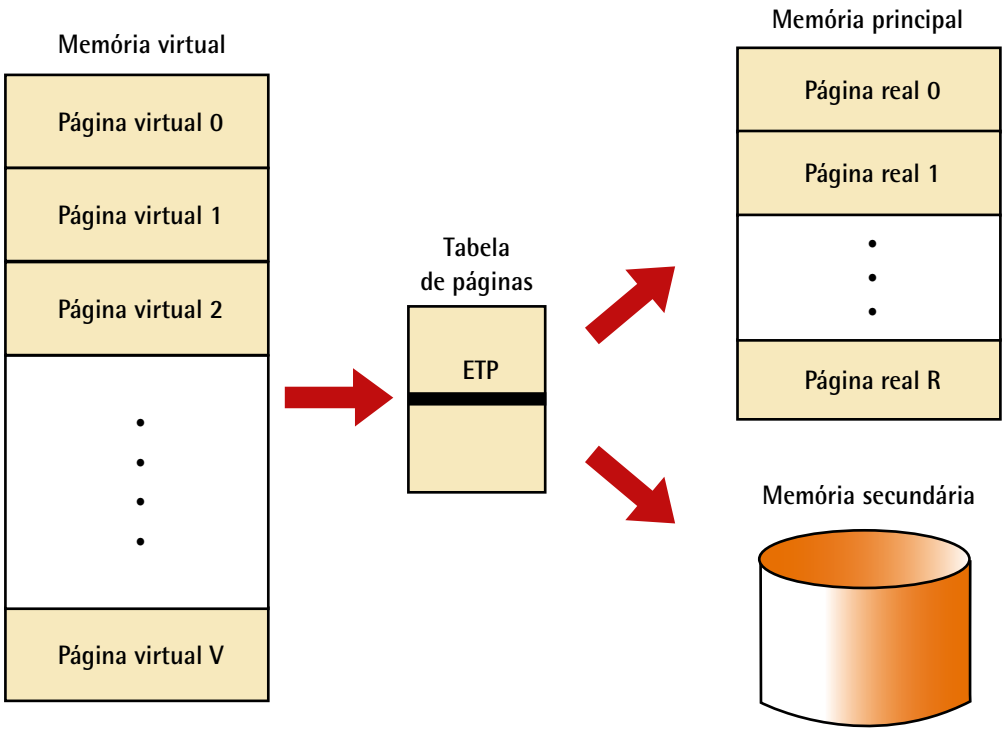

Figura 54 – Tabela de páginas

Fonte: Machado e Maia (2013, p. 164).

Blocos de tamanho fixo, por exemplo, 4 Kb são denominados de página. O espaço de endereçamento virtual é dividido em páginas virtuais. Mistura de tipo de dados.

Um dos campos principais da ETP é o bit de validade ou valid bit, que indica se uma página está ou não na memória principal. Se o bit for 0, não está carregada e é gerada uma interrupção de falta de página ou page fault, que desvia a execução para o sistema operacional. Neste caso o sistema transfere a página da memória secundária para a memória principal, realizando page in, ou paginação. Se o bit for igual a 1, a página está já carregada na memória principal.

Outros componentes da tabela são o número da página real ou *page frame number*, os bits de proteção (leitura, execução ou leitura/escrita), o bit de modificação, o bit de cache e o bit de referência.

A política de alocação de páginas define quantos frames cada processo poderá manter na memória principal. Há duas alternativas: alocação fixa ou alocação variável. Na alocação fixa, cada processo possui um número máximo de frames que pode ser utilizado na execução do programa. Caso o número de páginas reais seja insuficiente, uma página do processo em questão é descartada para que uma nova seja carregada. Na alocação variável, o número máximo de páginas alocadas varia dependendo da taxa de paginação e da ocupação da memória principal.

Através da política de busca de páginas, determina-se quando uma página deve ser carregada para a memória principal. Existem duas estratégias: paginação por demanda ou paginação antecipada. Na paginação por demanda, as páginas dos processos são transferidas da memória secundária para a memória principal somente quando referenciadas. Serão levadas somente aquelas necessárias à execução do programa.

Por meio da paginação antecipada, além da página referenciada, outras páginas que podem ou não ser necessárias ao processo ao longo do processamento são carregadas na memória principal.

A política de substituição de páginas determina como o sistema operacional seleciona qual página será liberada da memória principal para que outra a substitua. Existem duas políticas principais: política de substituição de páginas local e política de substituição de páginas global.

Na política de substituição de páginas local, sempre que um processo necessitar de uma nova página, o sistema selecionará uma do processo em questão para ser substituída. Já na política de substituição de páginas global, todas as páginas alocadas na memória principal são candidatas à substituição, independentemente do processo que a gerou.

5.4.2 Memória virtual por segmentação

Nesta técnica o espaço de endereçamento virtual é dividido em blocos de tamanhos diferentes e arbitrários chamados de segmentos. Um programa será dividido logicamente em sub-rotinas e estruturas de dados, que são alocadas em segmentos na memória principal.

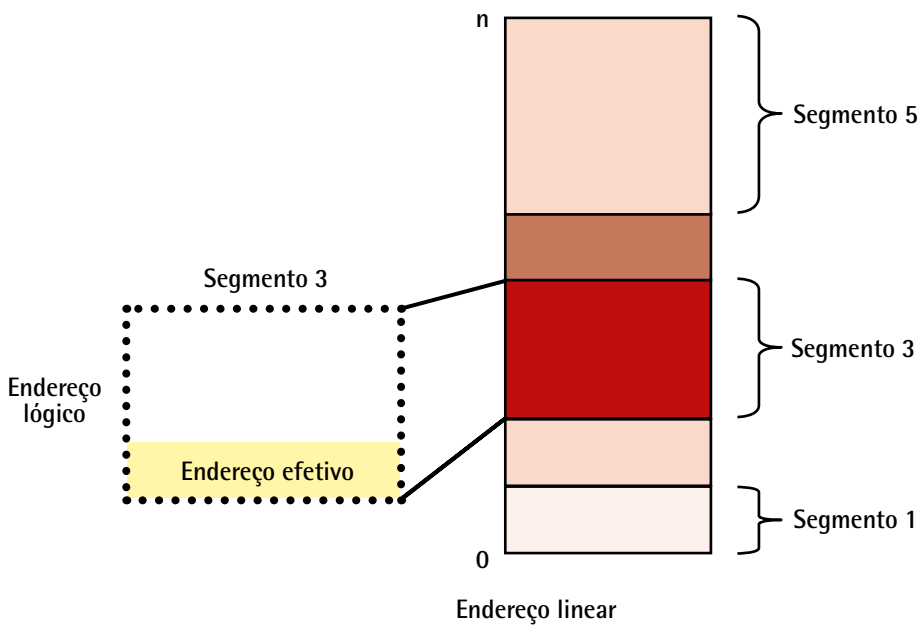

Figura 55 – Segmentação

Fonte: Furukawa e Nunes (2011, p. 67).

Na figura anterior, foram apresentados cinco segmentos de tamanhos e tipos diferenciados, em destaque o lado esquerdo, em que houve ainda a demonstração do segmento 3, que está parcialmente usado. O espaço de endereçamento virtual de um processo possui um número máximo de segmentos, podendo cada um deles ser de diferente grandeza com um limite de tamanho definido.

Uma vantagem desta técnica é a flexibilidade que permite que o tamanho do segmento seja alterável. Entretanto, a segmentação causa fragmentação externa, posto que há várias áreas livres na memória principal, mas nenhuma é suficiente para alocar um novo segmento.

5.4.3 Memória virtual por paginação X segmentação

O quadro a seguir compara as técnicas de memória virtual por paginação e por segmentação.

Quadro 5 – Características das técnicas de paginação e segmentação

Característica	Memória virtual por paginação	Memória virtual por segmentação
Tamanho dos blocos de memória	Igual	Diferente
Proteção	Complexa	Mais simples
Compartilhamento	Complexa	Mais simples
Estruturas de dados dinâmicas	Complexa	Mais simples
Fragmentação interna	Pode existir	Não existe
Fragmentação externa	Não existe	Pode existir
Programação modular	Dispensável	Indispensável
Alteração do programa	Mais trabalhosa	Mais simples

Adaptado de: Machado e Maia (2013, p. 184).

Visando extrair as vantagens de ambas as técnicas, foi desenvolvida a técnica de memória virtual por segmento com paginação. Nela, o espaço de endereçamento é dividido em segmentos e cada um deles será dividido em páginas.

Com isso, a aplicação será mapeada em segmento de tamanhos diferentes que dependem das sub-rotinas e estrutura de dados definidas no programa. O sistema, por sua vez, irá tratar cada segmento como um conjunto de páginas que possuem o mesmo tamanho. Cada segmento terá o mapeamento de uma das páginas à qual está associado.

Saiba mais

Existem alguns simuladores de sistema operacional no qual é possível entender os conceitos da gerência de memória virtual. O software SOsim, simulador para ensino de sistemas operacionais, foi desenvolvido pelo Prof. Luiz Paulo Maia:

MAIA, L. P. *SOsim*: simulador para o ensino de sistemas operacionais – Versão 2.0. Rio de Janeiro: UFRJ, 2007. Disponível em: https://bit.ly/2Qeb509. Acesso em: 14 mar. 2023.

6 SISTEMA DE ARQUIVOS

Desde os primeiros sistemas computacionais, ficou evidente a necessidade de armazenamento de informações para utilização futura, como programas e dados. O conceito de arquivo foi desenvolvido visando simplificar o armazenamento e a busca de informações e ambas são tarefas fundamentais para qualquer tipo de aplicação.

O sistema de arquivos é a parte mais visível de um sistema operacional; praticamente todos os usuários sabem manipular arquivos e a sua manipulação é frequente. Ele é composto de duas partes: uma coleção de arquivos, cada um deles armazenando dados relacionados, e uma estrutura de diretórios, responsável pela organização e fornecimento de informações sobre todos os arquivos no sistema.

Um processo deve ter a capacidade de leitura e gravação de forma permanente para um grande volume de dados em dispositivos como fitas e discos, e também pode compartilhar esses dados com outros processos.

6.1 Arquivos

De acordo com Maziero (2019, p. 278), "Um arquivo é essencialmente uma sequência de bytes armazenada em um dispositivo físico não volátil, como um disco rígido ou de estado sólido, que preserva seu conteúdo mesmo quando desligado". Os arquivos são gerenciados pelo sistema operacional de modo a facilitar o acesso dos usuários ao seu conteúdo e representam um armazenamento persistente. São constituídos por informações logicamente relacionadas e podem representar instruções ou dados.

Os processos podem ler ou escrever em arquivos ou ainda criar novos arquivos. As informações armazenadas em arquivos devem ser persistentes, ou seja, não podem ser influenciadas pela criação ou término de um processo, e um arquivo somente deverá desaparecer quando um usuário que tenha permissão apagá-lo.

Por meio de chamada de sistema ou system calls os processos e arquivos podem ser estruturados, nomeados, utilizados, protegidos e gerenciados.

Lembrete

Um arquivo pode ser entendido como uma sequência de bytes armazenada em um dispositivo físico não volátil, por exemplo, o disco rígido ou de estado sólido, e que mantenha seu conteúdo mesmo quando desligado.

6.1.1 Atributos de arquivos

Segundo Silberschatz, Galvin e Gagne (2015), os principais atributos de um arquivo são: nome, tipo, locação, tamanho, proteção, hora/data e identificação de usuário.

Quando um arquivo é criado, atribui-se um **nome**, que é a forma de identificação voltada para o usuário. Cada sistema de arquivos tem suas próprias regras de atribuição de nomes de arquivos, mas todos os sistemas operacionais atuais permitem strings de tamanhos limitados contendo algarismos e caracteres especiais, como nomes de arquivo válidos. Além do nome, o sistema de arquivos possui para cada arquivo um identificador, que é um rótulo exclusivo, usualmente um número bastante extenso. Em alguns sistemas de arquivos, é feita distinção entre caracteres alfabéticos maiúsculos e minúsculos.

A identificação de um arquivo é composta de duas partes separadas por um ponto: a parte antes do ponto é o nome do arquivo e a parte pós-ponto é chamada de **tipo** ou **extensão do arquivo**. Normalmente, trata-se de uma informação necessária para sistemas que suportam diferentes tipos de arquivo. O quadro a seguir lista alguns tipos de arquivo e a sua extensão.

Quadro 6 – Tipos de arquivo

Extensão	Tipo de arquivo
.pdf	Arquivo no formato *portable document format*
.exe	Arquivo executável
.jpg	Fotografia codifica com o padrão JPEG
.txt	Arquivo de texto geral
.c	Programa fonte em linguagem C

A **locação** corresponde a um ponteiro para um dispositivo e a locação do arquivo nesse dispositivo, isto é, indicação do dispositivo físico onde o arquivo se encontra e da posição do arquivo dentro dele. Os arquivos podem ser armazenados por diversos dispositivos físicos, tais como discos, fitas magnéticas, pen drives, entre outros.

Cada arquivo possui um **tamanho** que corresponde ao espaço ocupado em bytes, palavras ou blocos. Por exemplo, o sistema de arquivos informou que o arquivo relatorio.txt possui 11.349 bytes.

Outro atributo de um arquivo é **proteção**. As informações de controle de acesso determinam qual usuário pode ler, gravar, executar o arquivo. Os dados **data/hora** podem ser mantidos para criação, última modificação e última utilização de arquivo. Já **identificação de usuário** corresponde a quem criou o arquivo, quem modificou e o último que acessou. Essas informações podem ser úteis para proteção, segurança e monitoramento do uso.

Observação

Dependendo de sua natureza, alguns atributos, como data/hora de criação, não podem ser alterados. Por outro lado, tamanho e data/hora da última atualização são alterados pelo próprio sistema operacional. O usuário pode modificar alguns atributos do arquivo, tais como proteção, permissão de escrita/leitura e senha de acesso.

6.1.2 Organização de arquivos

A organização de arquivos determina como os dados são internamente armazenados, e tal estrutura depende do tipo de informação presente no arquivo. Por exemplo, a estrutura de um documento de texto é bastante diferente daquela de um arquivo executável. Dependendo da forma como o arquivo está organizado, o seu sistema poderá recuperá-lo de diversas maneiras.

A forma mais simples de organização de arquivos é através de uma **sequência não estruturada de bytes**, quando não há imposição de nenhuma estrutura lógica para os dados. Nesse caso, o controle dos critérios é definido pela aplicação. Uma vantagem desse método é a flexibilidade para criação de estrutura de dados.

Na operação com fitas magnéticas, uma das primeiras tecnologias para armazenamento, os dados são lidos na ordem em que foram gravados, e a gravação de novos registros somente é possível no final do arquivo. Caso contrário, dados anteriores poderiam ser sobrescritos. Esse tipo de acesso é denominado **acesso sequencial**, sendo característico para operações com a fita magnética como meio de armazenamento e que tenham uma elevada ineficiência para a leitura.

Com a introdução dos discos magnéticos, foram introduzidos outros métodos de acesso. O primeiro foi o **acesso direto**, no qual a leitura e gravação de um registro ocorrem diretamente na sua posição, sem ter de passar por outras, pois o arquivo é acessado através de seu número de registro, isto é, sua posição relativa ao início do arquivo. É eliminada a necessidade de leitura dos registros na mesma ordem de sua gravação. Uma limitação desse tipo de acesso é o fato de ser somente implementável com arquivos de tamanho fixo.

O **acesso indexado ou acesso por chave** foi criado tendo como base o acesso direto. Nele, o arquivo inclui uma área de índice que possui ponteiros para diversos registros. Quando uma aplicação necessita acessar determinado registro, ela especifica a chave a ser utilizada para pesquisa na área do índice, o que indicará o ponteiro adequado. Com posse dessa informações, ocorre o acesso direto ao registro resultante.

6.1.3 Operações de entrada de saída

O sistema de arquivos disponibiliza um conjunto de rotinas que permitem às aplicações realizar operações de E/S, como tradução de nomes em endereços, leitura e gravação de dados criação e eliminação de arquivos. O acesso aos arquivos é executado através de um conjunto de operações implementadas por meio de chamadas de sistemas e funções de bibliotecas.

As rotinas de E/S têm como função disponibilizar uma interface simples e uniforme entre a aplicação e os diversos dispositivos. O quadro a seguir lista algumas das operações com arquivos.

Quadro 7 – Operação com arquivos

Rotina	Descrição	Atributos
CREATE	Criação de arquivos	Informações de controle de cada arquivo. Variam em função do sistema de arquivos, porém eles estão presentes em quase todos os sistemas: tamanho, proteção, identificação do criador, data da criação
OPEN	Abertura de um arquivo	
READ	Leitura de um arquivo	
WRITE	Gravação de um arquivo	
CLOSE	Fechamento de um arquivo	
DELETE	Eliminação de um arquivo	

6.2 Sistema de arquivos

O sistema de arquivos consiste na organização de arquivos e diretórios de um dispositivo físico. Ele pode ser considerado como uma imensa estrutura de dados armazenada de forma persistente no dispositivo físico.

Há uma grande quantidade de sistemas de arquivos, entre os quais podem ser citados o NTFS, para os sistemas Windows; Ext2/Ext3/Ext4, para sistemas Linux; HPFS, para MacOS, FFS, utilizado em sistemas Solaris; e o sistema de arquivos FAT, que é utilizado em pen drives USB, câmeras fotográficas digitais e leitores MP3.

O mesmo dispositivo físico pode ser estruturado em uma ou mais partições ou volumes. Para cada partição, pode existir um sistema de arquivos próprio. Por exemplo, um computador com duas partições, uma executando Windows e a outra Linux, terá dois sistemas de arquivos.

Podemos estabelecer dois pontos de vista para o sistema de arquivos: ponto de vista do usuário ou alto nível e ponto de vista do sistema operacional ou baixo nível.

Olhando sob a perspectiva do usuário, os aspectos que impactam são a interface e a forma como os arquivos aparecem no sistema operacional, as operações que o usuário pode realizar e as regras de nomeação de arquivos e de proteção.

Por outro lado, o sistema operacional verifica como os arquivos são armazenados fisicamente e como são referenciados, através de links para chamar um arquivo de outro diretório.

Para um sistema de arquivos, deve-se considerar que cada arquivo é composto de dados e metadados. Os dados de um arquivo representam o seu conteúdo em si, por exemplo, um documento de texto, uma música, uma planilha. Por outro lado, os metadados do arquivo representam seus atributos, tais como nome, localização do arquivo, identificador do usuário que criou o arquivo, permissões de acesso, data de criação e modificação etc. Também são considerados metadados as informações de controle necessárias para localizar e manter seu conteúdo no disco e aquelas essenciais à gestão do sistema de arquivos, como os mapas de blocos livres.

6.3 Diretório

O sistema operacional organiza logicamente os diversos arquivos contidos em um disco através da estrutura de diretórios. O **diretório** é uma estrutura de dados que possui entradas associadas aos arquivos em que cada registro guarda informações como nome, localização física, organização e demais atributos.

A forma mais elementar de implementação de uma estrutura de diretórios é denominada **nível único** ou *single-level directory*. Nela, há apenas um diretório que contém todos os arquivos do disco, como mostra a figura a seguir. Essa estrutura é bastante limitada e não permite que usuários criem arquivos com o mesmo nome, pois causaria um conflito no acesso aos arquivos.

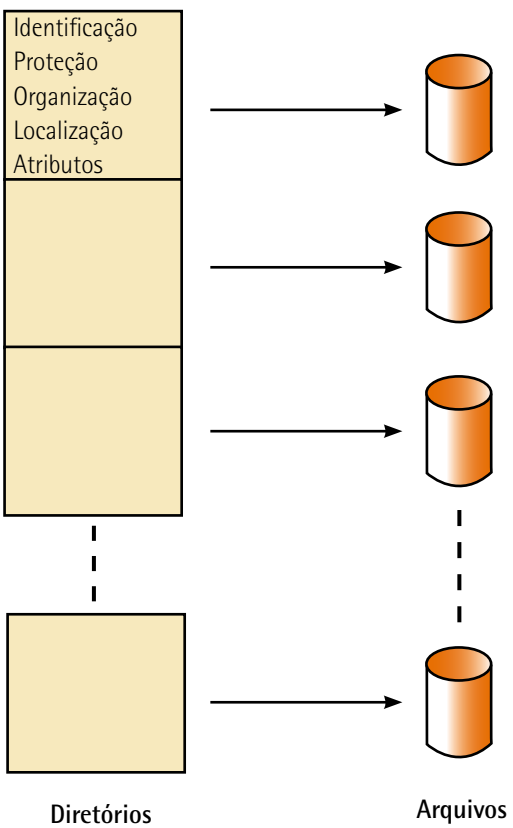

Figura 56 – Estrutura de diretório de nível único

Fonte: Machado e Maia (2013, p. 198).

Partindo desse modelo, foi implementada uma estrutura em que cada usuário possui um diretório particular denominado *user file directory* (UFD). Com isso, cada usuário pode criar arquivos com qualquer nome, sem precisar conhecer os demais do disco. Para a localização desses arquivos nessa estrutura, foi construído um nível de diretório adicional denominado *master file directory* (MFD). Ele serve para controlar os diretórios individuais dos usuários e é indexado pelo nome do usuário. A figura a seguir apresenta a estrutura de diretórios com dois níveis ou *two-level directory*.

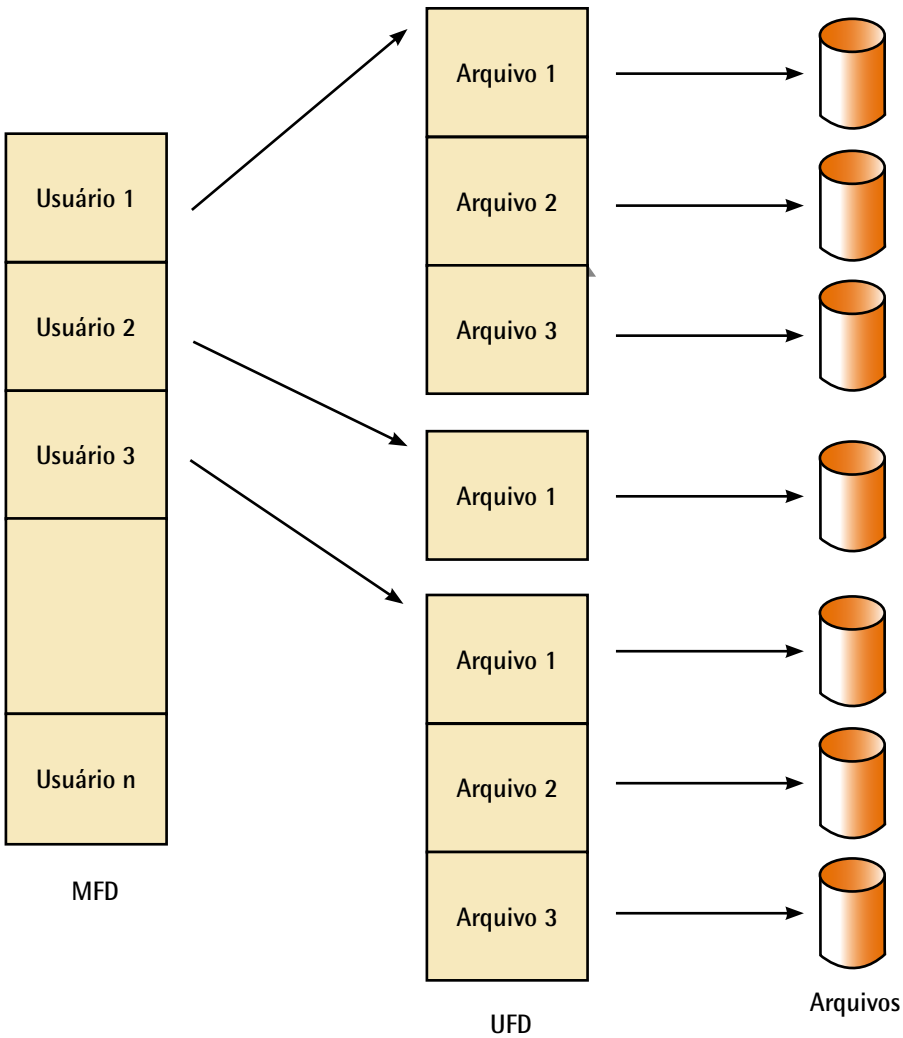

Figura 57 – Estrutura de diretório de dois níveis

Fonte: Machado e Maia (2013, p. 198).

Com extensão do modelo de dois níveis, foi desenvolvida uma estrutura que permite múltiplos níveis, visando que os arquivos fossem logicamente melhor organizados. Essa nova implementação é chamada estrutura de diretórios em árvore ou *tree-structured directory* e a maioria dos sistemas operacionais adota esse modelo.

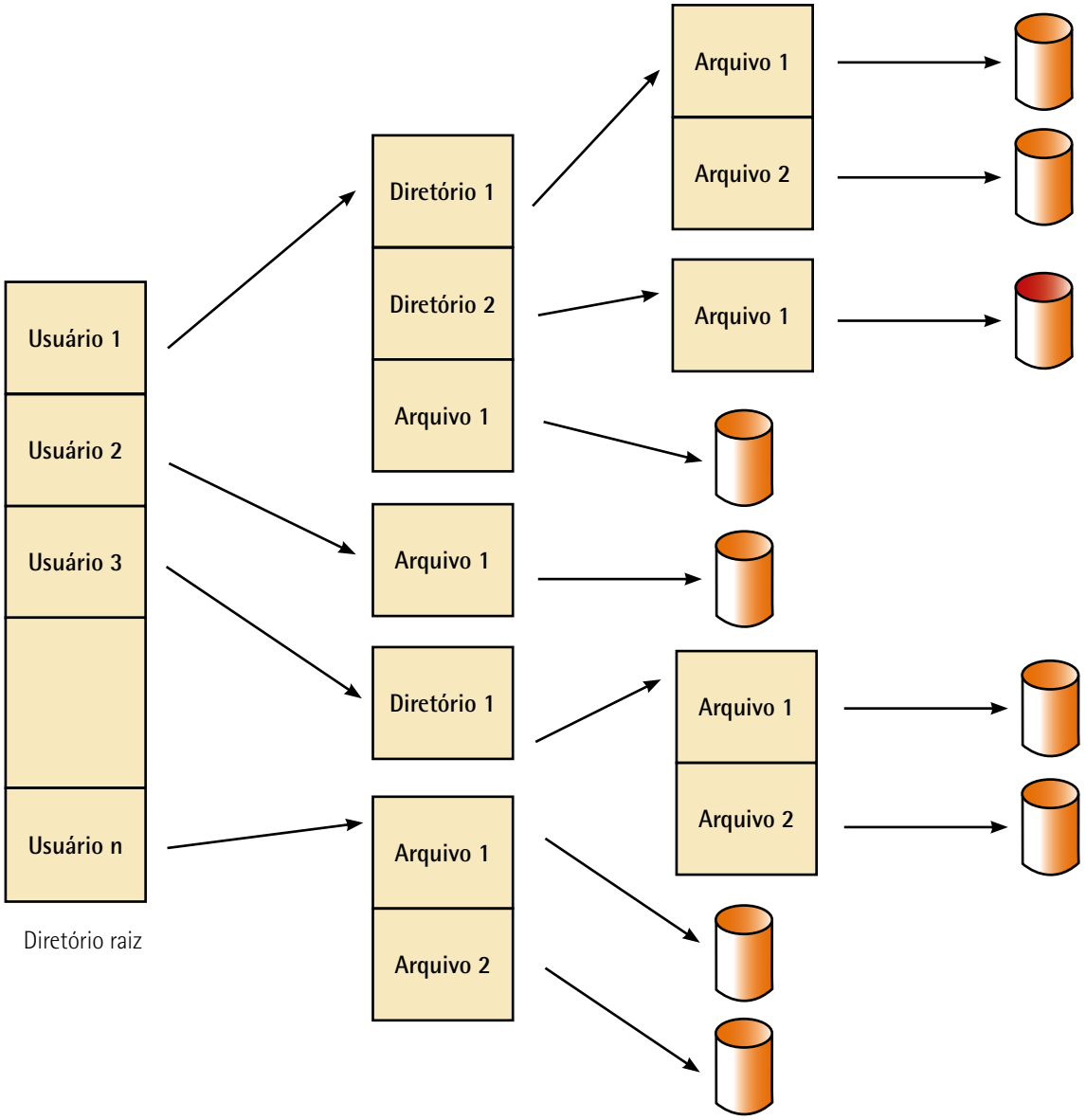

Figura 58 – Estrutura de diretório em árvore

Fonte: Machado e Maia (2013, p. 200).

Saiba mais

A fim de conhecer os diferentes sistemas de arquivos existentes no Linux, leia o seguinte artigo:

MATEUS. File systems: Comparando os sistemas de arquivos do Linux. *Revista Infra Magazine*, v. 18, 2014. Disponível em: https://bit.ly/403m8fo. Acesso em: 14 mar. 2023.

6.4 Gerência de espaço livre

A fim de serem criados novos arquivos em disco, o sistema operacional precisa ter o controle de quais áreas ou blocos no disco estão livres. Para cada sistema existe alguma estrutura de dados, por exemplo, uma lista ou tabela, responsável por armazenar informações para gerenciamento do espaço livre do disco pelo sistema de arquivos.

A forma mais simples de implementar uma estrutura de espaços livres é através de uma tabela denominada **mapa de bits** (bitmap). Cada entrada nela é associada a um bloco do disco representado por um bit, podendo assumir valor igual a 0 (bloco livre) ou 1 (bloco ocupado).

A segunda forma é encadear todos os blocos livres do disco, sendo que cada um deles possui uma área para armazenar o endereço do próximo bloco.

Outra solução leva em consideração que blocos contíguos são geralmente alocados e liberados simultaneamente, mantendo uma tabela com o endereço do 1º bloco e o número de blocos livres contíguos que seguem.

6.5 Gerência de alocação do espaço livre

Na **alocação contígua**, o sistema de arquivos armazena um arquivo em blocos sequenciais dispostos no disco. Ele localiza um arquivo através do endereço do 1º bloco e da sua extensão em blocos.

Seu principal problema é a alocação de espaço livre para novos arquivos. Caso um arquivo deva ser criado com um determinado tamanho, é necessário existir uma quantidade suficiente de blocos contíguos no disco para a realização de alocação.

Existem estratégias de alocação para selecionar qual segmento na lista de blocos será alocado:

- **First-fit**: o primeiro segmento livre com tamanho suficiente é alocado.
- **Best-fit**: o menor segmento livre disponível com tamanho suficiente para armazenar o arquivo é selecionado. A busca em toda a lista se faz necessária para a seleção do segmento.
- **Worst-fit**: o maior segmento é alocado.

Para qualquer uma dessas estratégias, a alocação contígua apresenta uma dificuldade que é conhecida como fragmentação dos espaços livres. Considerando que arquivos são criados e apagados frequentemente, os segmentos têm a tendência de se fragmentar em pedaços cada vez menores por todo o disco. A situação torna-se crítica quando existem blocos livres em um disco, mas não há um segmento contíguo no qual o arquivo possa ser alocado por completo.

Para solucionar esse problema, é utilizada a desfragmentação, que consiste em uma rotina que reorganiza todos os arquivos no disco de forma a deixar apenas um segmento de blocos livres. Essa operação consome um grande período e possui efeito temporário, por isso necessita ser realizada de forma periódica.

Resumo

Nesta unidade, vimos que para sistemas multiprogramados existem formas de alocação de memória determinadas pelo gerenciamento de memória. Eventualmente, é necessário que o sistema operacional utilize mais memória principal ou memória física, o que exige a troca de processos entre a memória principal e o disco. Esse processo é denominado swapping.

Ainda foram apresentados os conceitos de memória virtual e as suas formas de organização: por paginação, por segmentação ou por segmentação com paginação.

Comentamos acerca do fato de o sistema de arquivos ser um aspecto bastante visível para os usuários dos sistemas operacionais e existirem diversos atributos para um arquivo, como nome, tipo, data/hora de criação e de atualização, propriedade de acesso, localização, entre outros.

Por fim, observamos que cada sistema operacional possui sistema de arquivos específico que contempla a estrutura de diretórios.

Exercícios

Questão 1. (Idecan 2021, adaptada) No contexto da arquitetura de computadores, observe a figura a seguir, relacionada à hierarquia de memória dos microcomputadores.

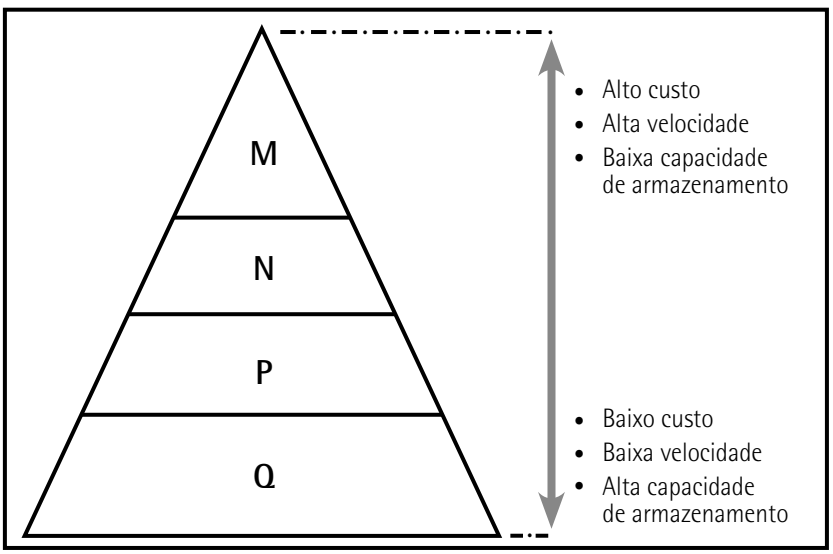

Figura 59

Comparando-se os diferentes tipos, podemos afirmar que M, N, P e Q representam, respectivamente, as memórias:

A) Pipeline, registradores, cache e principal.

B) Registradores, cache, principal e secundária.

C) Cache, principal, secundária e pipeline.

D) Principal, secundária, pipeline e registradores.

E) Secundária, cache, registradores e principal.

Resposta correta: alternativa B.

Análise da questão

Em um sistema computacional moderno, a memória mais rápida, mais cara e com menor capacidade de armazenamento é o conjunto de registradores que integram o chip da unidade central de processamento. Assim, na figura da hierarquia de memória, M representa os registradores, que se encontram no topo da pirâmide.

Logo abaixo deles, temos as memórias cache. Essas memórias são um pouco mais baratas de serem produzidas do que os registradores e possuem capacidade de armazenamento maior, porém são mais lentas. Portanto, N representa as memórias cache do sistema computacional.

Logo abaixo das memórias cache, há a memória principal do sistema, que chamamos popularmente de memória RAM. Ela usa uma tecnologia de fabricação baseada em capacitores, o que permite uma fabricação mais barata e mais capacidade de armazenamento, quando comparada às memórias cache. Porém, como esperado, isso também vem a um custo de velocidade, já que a memória principal é mais lenta do que a cache. Portanto, P representa a memória principal.

Por fim, na base da pirâmide, consta a memória secundária, que consiste no conjunto de dispositivos de armazenamento persistente ou não volátil, ou seja, que não perdem as informações armazenadas mesmo após a interrupção da alimentação elétrica. Desse modo, mesmo após desligar o sistema computacional, as informações continuam armazenadas, estando disponíveis quando o sistema for religado. A memória secundária é aquela que apresenta maior capacidade de armazenamento e é a mais barata de ser produzida. No entanto, ela é o tipo de memória mais lento do sistema computacional. Temos, portanto, Q representando a memória secundária.

Questão 2. (Ufac 2022, adaptada) Leia o texto a seguir, a respeito de sistemas de arquivos.

Um sistema de arquivos é um conjunto de estruturas lógicas, ou seja, feitas diretamente via software, que permite ao sistema operacional ter acesso e controlar os dados gravados no disco. Cada sistema operacional tem um sistema de arquivos nativo diferente, e cada sistema de arquivos possui as suas peculiaridades, como limitações, qualidade, velocidade, gerenciamento de espaço, entre outras características. É o sistema de arquivos que define como os bytes que compõem um arquivo serão armazenados no disco e de que forma o sistema operacional terá acesso aos dados.

Adaptado de: http://glo.bo/40suRHY. Acesso em: 14 mar. 2023.

Nesse contexto, assinale a alternativa que apresenta o sistema de arquivos mais comumente utilizado no sistema operacional Microsoft Windows 10.

A) FAT.

B) LVM.

C) exFAT.

D) APFS.

E) NTFS.

Resposta correta: alternativa E.

Análise da questão

Na época do Windows 95, a Microsoft usava o sistema nativo de arquivos de disco FAT16 (File Allocation Table de 16 bits). Devido às suas limitações, ele foi substituído pelo FAT32, que, anos depois, foi trocado pelo NTFS (de Windows NT File System). O sistema NTFS é usado até hoje e, portanto, é o sistema de arquivos de disco mais comumente utilizado no Windows 10, sendo seu sistema nativo atualmente.

No entanto, muitos sistemas de arquivos estão em uso hoje em dia, e a maioria dos sistemas operacionais atuais suporta mais de um. O Windows suporta os formatos de sistemas de arquivos de disco FAT, FAT32 e NTFS, assim como formatos de sistemas de arquivos em CD-ROM e DVD.