Introdução a Arquitetura de Sistemas Operacionais - Parte 02

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Julho / 2017

Nota

- Com exceção das figuras que possuem referência bibliográfica, todo o restante do conteúdo destes *slides* foram extraídos do livro dos autores “Francis Berenger Machado” e “Luiz Paulo Maia”. Todos referenciados no último *slide* desta apresentação.

Processos do Sistema Operacional

- O conceito de processo, além de estar associado a aplicações de usuários, pode também ser implementado na própria arquitetura do sistema operacional.

- Quando processos são utilizados para a implementação de serviços do sistema, retira-se código de seu núcleo, tornando-o menor e mais estável.

Processos do Sistema Operacional

- Alguns serviços que o sistema operacional pode implementar através de processos:
  - auditoria e segurança;
  - serviços de rede;
  - contabilização do uso de recursos;
  - contabilização de erros;
  - gerência de impressão;
  - gerência de *jobs batch*;
  - temporização;
  - comunicação de eventos;
  - interface de comandos (*shell*).

Gerência de Processador

- Com o surgimento dos sistemas multiprogramáveis, onde múltiplos processos poderiam permanecer na memória principal compartilhando o uso da CPU, a gerência do processador tornou-se uma das atividades mais importantes em um sistema operacional;

- A partir do momento em que diversos processos podem estar no estado de pronto, devem ser estabelecidos critérios para determinar qual processo será escolhido para fazer uso do processador;

- Os critérios utilizados para esta seleção compõem a chamada política de escalonamento, que é a base da gerência do processador e da multiprogramação em um sistema operacional.

Gerência de Processador

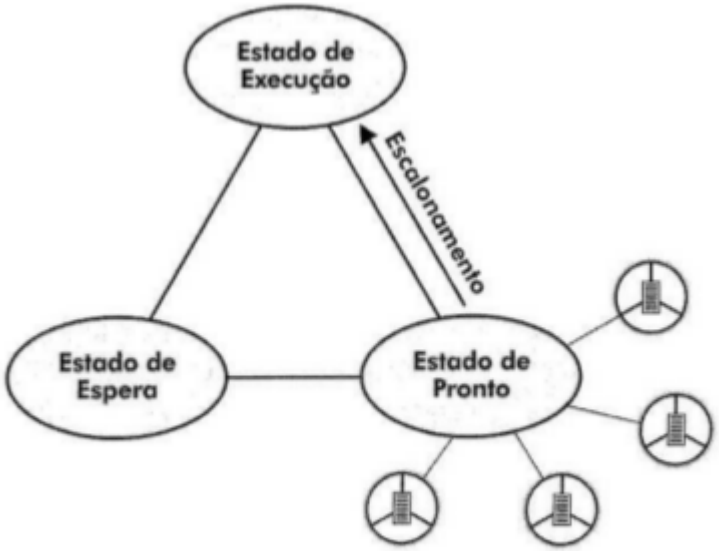


Figura 8.1: Escalonamento

Critérios de Escalonamento

- ***Throughput***: representa o número de processos executados em um determinado intervalo de tempo. Quanto maior o *throughput*, maior o número de tarefas executadas em função do tempo. A maximização do *throughput* é desejada na maioria dos sistemas;

- **Tempo de processador**: é o tempo que um processo leva no estado de execução durante seu processamento. As políticas de escalonamento não influenciam o tempo de processador de um processo, sendo este tempo em função apenas do código da aplicação e da entrada de dados;

- **Tempo de espera**: é o tempo total que um processo permanece na fila de pronto durante seu processamento, aguardando para ser executado.

Critérios de Escalonamento

- **Tempo de *turnaround***: é o tempo que um processo leva desde a sua criação até ao seu término, levando em consideração todo o tempo gasto na espera para alocação de memória, espera na fila de pronto, processamento na CPU e na fila de espera, como nas operações de E/S;

- **Tempo de resposta**: é o tempo decorrido entre uma requisição ao sistema ou à aplicação e o instante em que a resposta é exibida. Em sistemas interativos, podemos entender tempo de resposta como o tempo decorrido entre a última tecla digitada pelo usuário e o início da exibição do resultado no monitor.

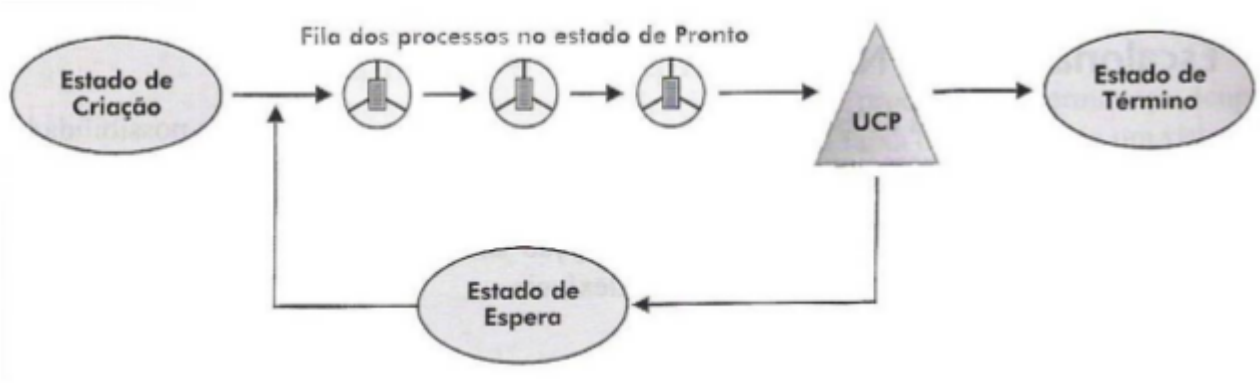
Escalonamentos Preemptivos e Não-preemptivos

- O escalonamento preemptivo é caracterizado pela possibilidade do sistema operacional interromper um processo em execução e passá-lo para o estado de pronto, com o objetivo de alocar outro processo na CPU;

- No escalonamento não-preemptivo, quando um processo está em execução nenhum evento externo pode ocasionar a perda do uso do processador. O processo somente sai do estado de execução caso termine seu processamento ou execute instruções do próprio código que ocasionem uma mudança para o estado de espera.

Algoritmos de Escalonamento

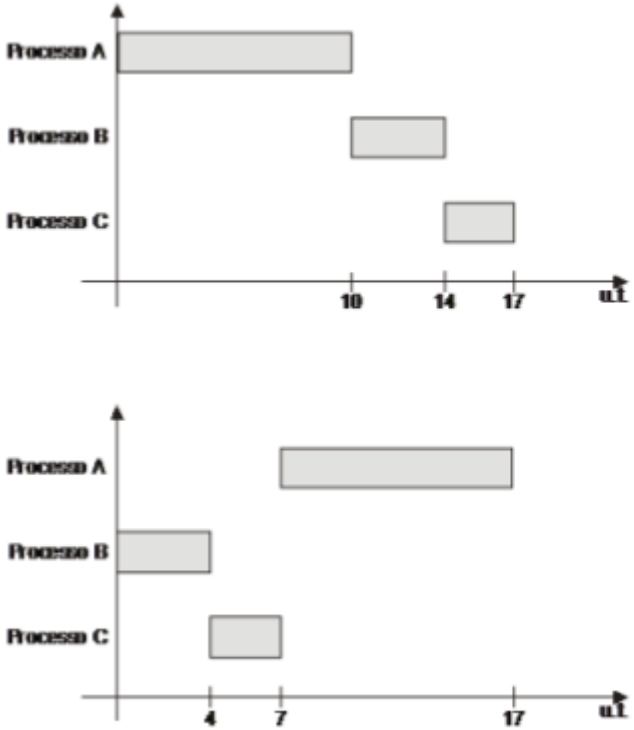
- FIFO (*First In, First Out*)
- O escalonamento FIFO é do tipo não-preemptivo e foi inicialmente implementado em sistemas monoprogramáveis com processamento *batch*, sendo ineficiente se aplicado na forma original em sistemas interativos de tempo compartilhado.



Cálculo do Tempo Médio de Espera

(a)

(b)



Processo	Tempo de processador (u.t.)
A	10
B	4
C	3

Figura 8.3: Exemplos de escalonamento FIFO

Escalonamento Menor *Job* Primeiro

- No escalonamento *Shortest-Job-First* (SJF *scheduling*), também conhecido como *Shortest-Process-Next* (*SPN scheduling*), o algoritmo de escalonamento seleciona o processo que tiver o menor tempo de processador ainda por executar. Dessa forma, o processo em estado de pronto que necessitar de menos tempo de CPU para terminar seu processamento é selecionado para execução.

Cálculo do Tempo Médio de Espera

- Calcule o tempo médio de espera, para os processos já vistos, utilizando o SJF.

Escalonamento pela Próxima Taxa de Resposta Mais Alta

- O escalonamento pela Próxima Taxa de Resposta Mais Alta (*Highest-Response-Ration-Next* – HRN) corrige algumas das fraquezas no SJF, particularmente o preconceito excessivo contra tarefas longas e o excessivo favoritismo para novas tarefas curtas;

- Prioridades dinâmicas são calculadas de acordo com a fórmula:
  - prioridade = (tempo de espera + tempo de serviço) / tempo de serviço
  -

Cálculo do Tempo Médio de Espera

- Calcule o tempo médio de espera, para os processos já vistos, utilizando o HRN.

Escalonamento Cooperativo

- A principal característica do escalonamento cooperativo está no fato da liberação do processador ser uma tarefa realizada exclusivamente pelo processo em execução, que de uma maneira cooperativa libera a CPU para um outro processo;

- Neste mecanismo, o processo em execução verifica periodicamente uma fila de mensagens para determinar se existem outros processos na fila de pronto;

- Como a interrupção do processo em execução não é responsabilidade do sistema operacional, algumas situações indesejadas podem ocorrer.

Escalonamento Circular

- Também conhecido como *Round Robin*

- É um escalonamento do tipo **preemptivo**, projetado especialmente para sistemas de tempo compartilhado. É bastante semelhante com o FIFO, porém, quando um processo passa para o estado de execução, existe um tempo limite para o uso contínuo do processador, denominado fatia de tempo (*time slice*) ou quantum;

- Neste escalonamento, toda vez que um processo é escalonado para execução, uma nova fatia de tempo é concedida. Caso a fatia de tempo expire, o sistema operacional interrompe o processo em execução, salva seu contexto e direciona-o para o final da fila de pronto;

- Esse mecanismo é conhecido por preempção por tempo.

Escalonamento Circular

- O valor da fatia de tempo depende da arquitetura de cada sistema operacional e, em geral, varia de 10 a 100 milissegundos. Este valor afeta diretamente o desempenho da política de escalonamento circular.

- Caso a fatia de tempo tenha um valor muito alto, este escalonamento tenderá a ter o mesmo comportamento do escalonamento FIFO.

- Caso o valor da fatia de tempo seja pequeno, a tendência é que haja um grande número de preempções, o que ocasionaria excessivas mudanças de contexto, prejudicando o desempenho do sistema e afetando o tempo de *turnaround* dos processos.

Escalonamento Circular

Figura 8.5: Escalonamento Circular

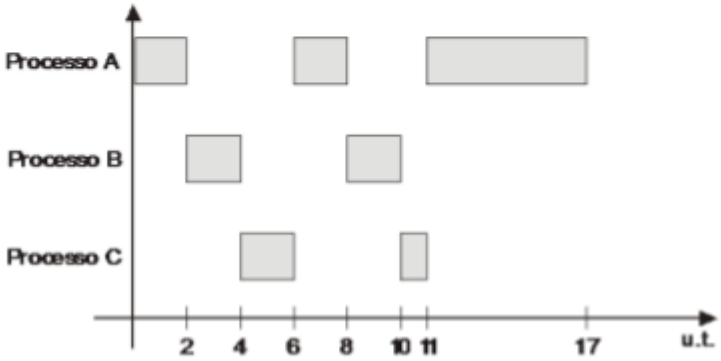


Figura 8.6: Exemplo de escalonamento circular

Escalonamento com Prioridade

- É um escalonamento do tipo **preemptivo** realizado com base em um valor associado a cada processo denominado prioridade de execução. O processo com maior prioridade no estado de pronto é sempre o escolhido para execução, e processos com valores iguais são escalonados seguindo o critério do FIFO;

- Neste escalonamento, o conceito de fatia de tempo não existe; ou seja, um processo em execução não pode sofrer preempção por tempo;

- No escalonamento por prioridade, a perda do uso do processador só ocorrerá no caso de uma mudança voluntária para o estado de espera ou quando um processo de prioridade maior passa para o estado de pronto.

Escalonamento com Prioridade

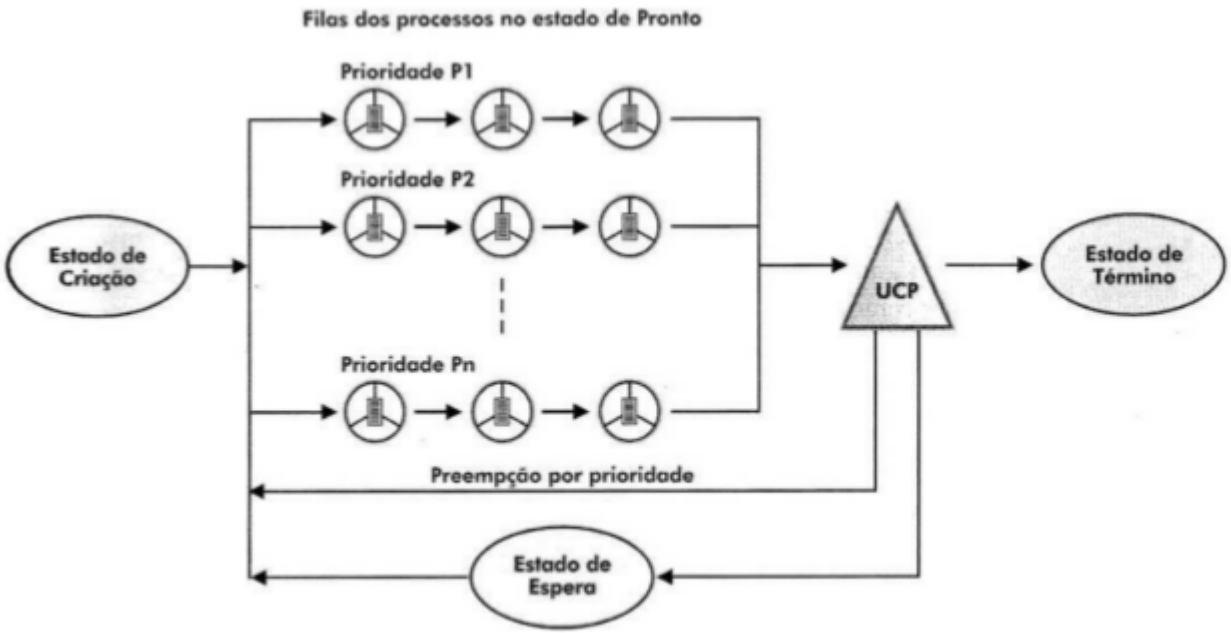
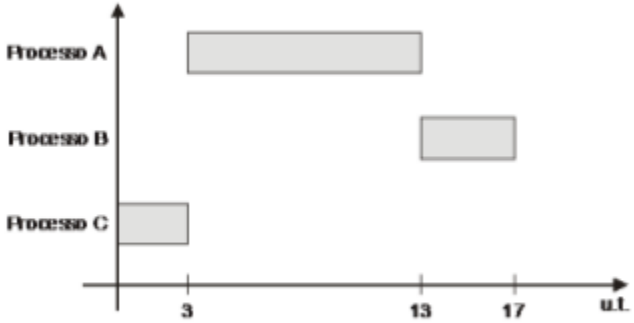


Figura 8.8: Escalonamento por prioridades

Escalonamento com Prioridade

A figura 8.9 exemplifica o escalonamento por prioridades com três processos de prioridades diferentes, onde 5 é o mais alto valor de prioridade.



Processo	Tempo de processador (u.t.)	Prioridade
A	10	2
B	4	1
C	3	3

Figura 8.9: Exemplo de escalonamento por prioridades

Escalonamento Circular Com Prioridade

- Este escalonamento implementa o conceito de fatia de tempo e de prioridade de execução associada a cada processo. Assim um processo permanece no estado de execução até que termine seu processamento, voluntariamente passe para o estado de espera ou sofra uma preempção por tempo ou prioridade.

Escalonamento Circular Com Prioridade

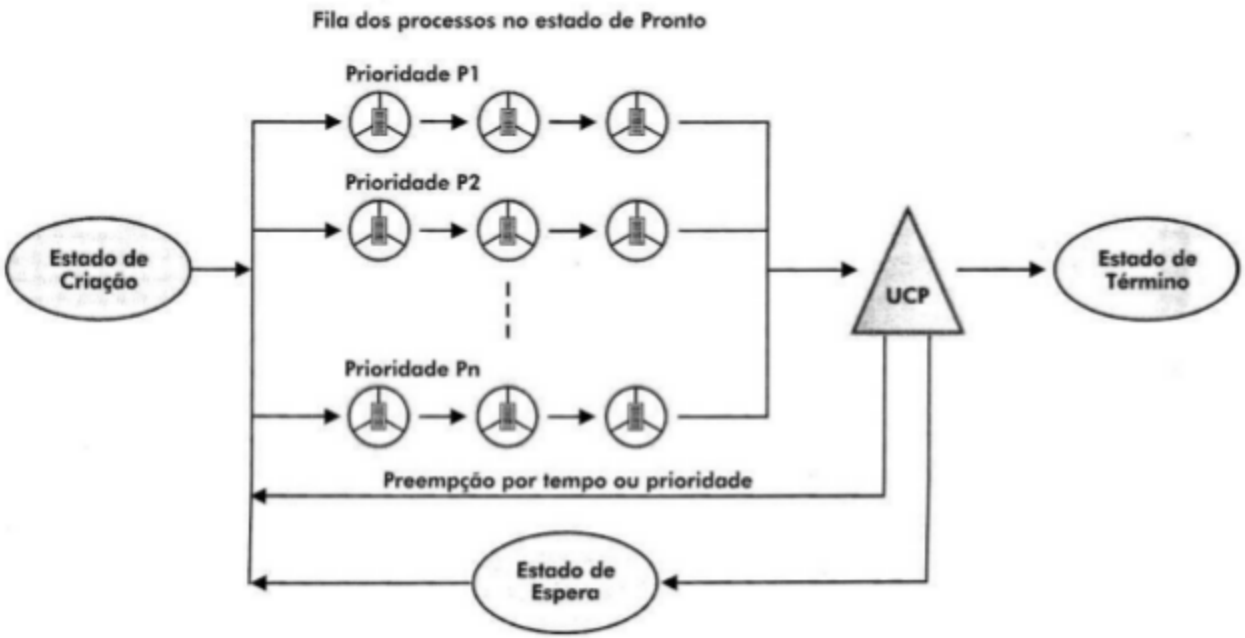


Figura 8.10: Escalonamento circular com prioridades

Escalonamento em Sistemas de Tempo Real

- Diferentemente dos sistemas de tempo compartilhado, onde a aplicação não é prejudicada pela variação no tempo de resposta, algumas aplicações específicas exigem respostas imediatas para a execução de determinadas tarefas;

- O escalonamento em sistemas de tempo real deve levar em consideração a importância relativa de cada tarefa na aplicação. Em função disso, o escalonamento por prioridades é o mais adequado, já que para cada processo uma prioridade é associada em função da importância do processo dentro da aplicação.

Gerência de Memória

- A memória principal sempre foi vista como um recurso escasso e caro. Uma das maiores preocupações dos projetistas foi desenvolver sistemas operacionais que não ocupassem muito espaço de memória e que otimizassem a utilização dos recursos computacionais;

- 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.

Gerência de Memória

- Enquanto nos sistemas monoprogramáveis a gerência da memória não é muito complexa, nos sistemas multiprogramáveis essa gerência se torna crítica, devido à necessidade de se maximizar o número de usuários e aplicações utilizando eficientemente o espaço da memória principal.

Gerência de Memória

- Em um ambiente de multiprogramação, o sistema operacional deve proteger as áreas de memória ocupadas por cada processo, além da área onde reside o próprio sistema.

- Caso um programa tente realizar algum acesso indevido à memória, o sistema de alguma forma deve impedi-lo.

Alocação Contígua Simples

- Foi implementada nos primeiros sistemas operacionais, porém ainda está presente em alguns sistemas monoprogramáveis;

- Nesse tipo de organização, a memória principal é subdividida em duas áreas: uma para o sistema operacional e outra para o programa do usuário.

Alocação Contígua Simples

Técnica de Overlay

- Na alocação contígua simples, todos os programas estão limitados ao tamanho da área de memória principal disponível para o usuário. Uma solução encontrada para o problema é dividir o programa em módulos, de forma que seja possível a execução independente de cada módulo, utilizando uma mesma área de memória;

- Considere um programa que tenha três módulos: um principal, um de cadastramento e outro de impressão, sendo os módulos de cadastramento e de impressão independentes. A independência do código significa que quando um módulo estiver na memória para execução, o outro não precisa necessariamente estar presente;

- O módulo principal é comum aos dois módulos; logo, deve permanecer na memória durante todo o tempo da execução do programa.

Técnica de *Overlay*

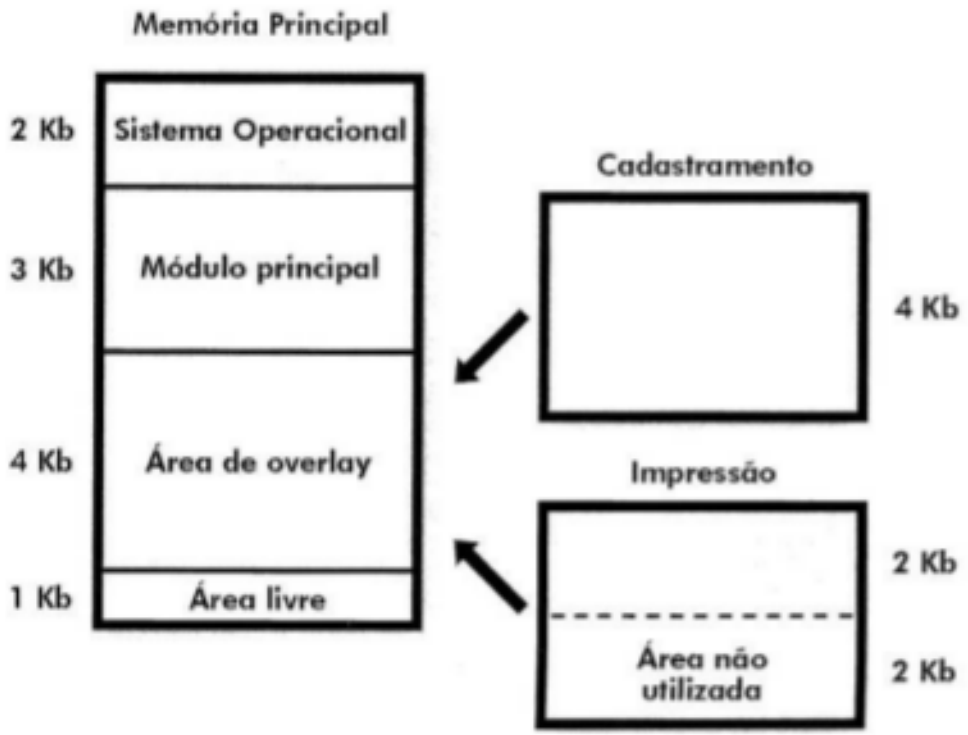


Figura 9.4: Técnica de overlay

Alocação Particionada Estática

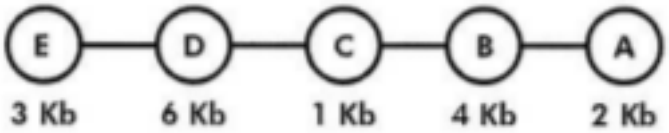
- Nos primeiros sistemas multiprogramáveis, a memória era dividida em pedaços de tamanho fixo, chamados partições;
- O tamanho das partições, estabelecido na fase de inicialização do sistema, era definido em função do tamanho dos programas que executariam no ambiente;
- Sempre que fosse necessária a alteração do tamanho de uma partição, o sistema deveria ser desativado e reinicializado com uma nova configuração.

Alocação Particionada Estática

Tabela de partições

Partição	Tamanho
1	2 Kb
2	5 Kb
3	8 Kb

Programas a serem executados:



Memória Principal

Figura 9.9: Alocação particionada estática

Alocação Particionada Estática

Problema:

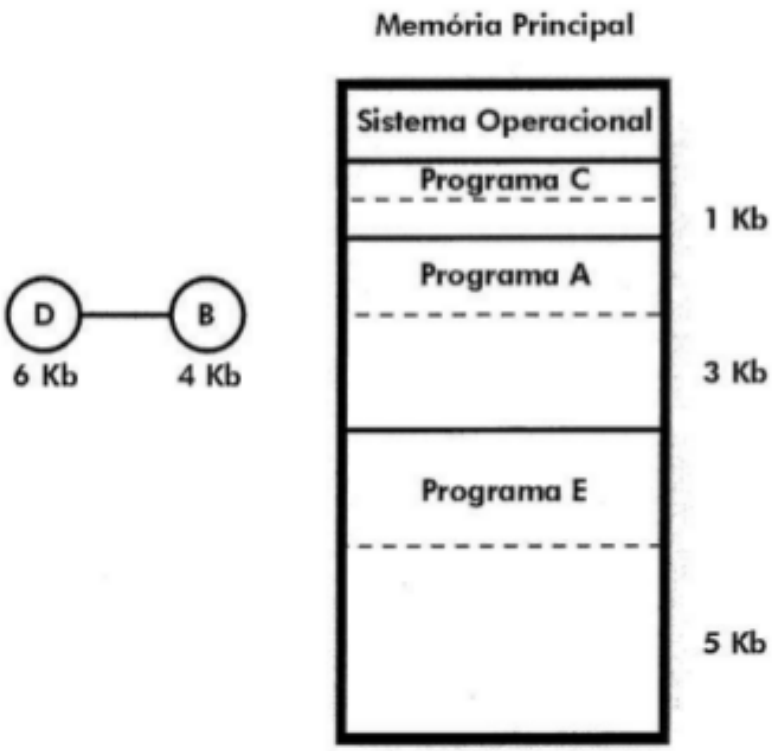


Figura 9.14: Fragmentação interna

Alocação Particionada Dinâmica

- Nesse esquema, cada programa utiliza o espaço necessário, tornando essa área sua partição;

- Como os programas utilizam apenas o espaço de que necessitam, no esquema de alocação particionada dinâmica o problema da fragmentação interna não ocorre.

Alocação Particionada Dinâmica

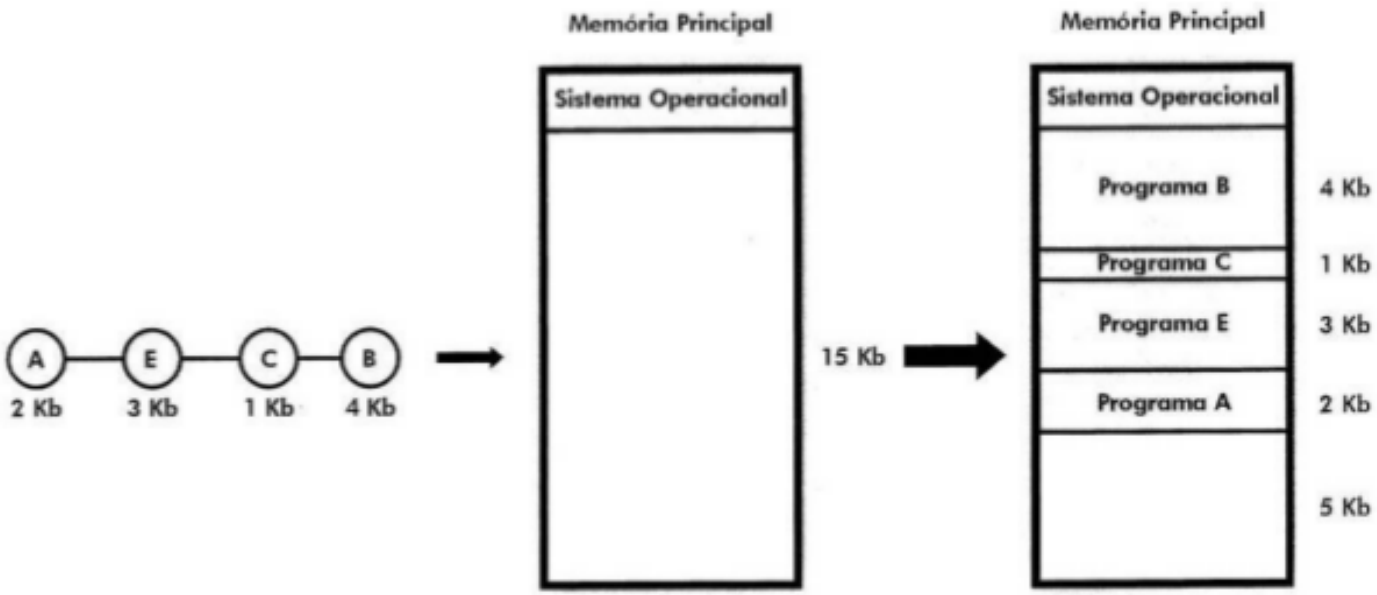


Figura 9.15: Alocação particionada dinâmica

Alocação Particionada Dinâmica

Problema:

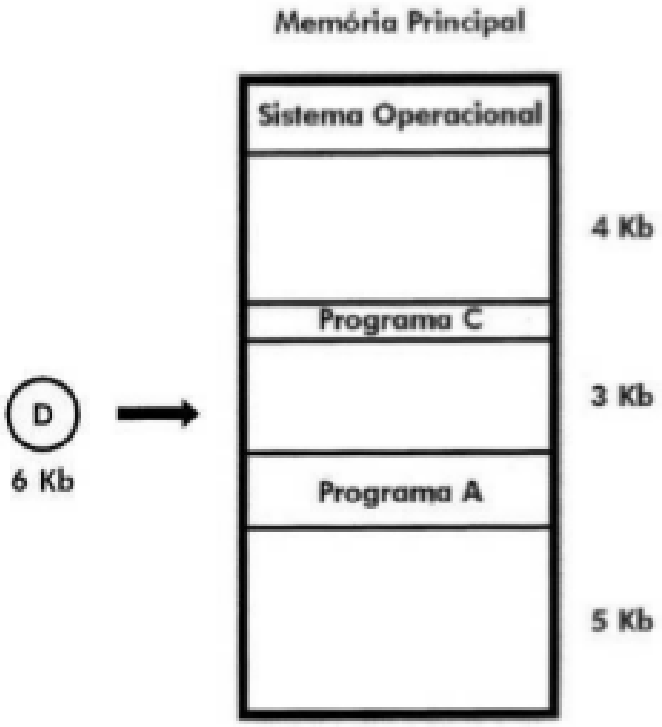


Figura 9.16: Fragmentação externa

Swapping

- Mesmo com o aumento da eficiência da multiprogramação e da gerência de memória, muitas vezes um programa não podia ser executado por falta de uma partição livre disponível;

- A técnica de *swapping* foi introduzida para contornar este problema.

Swapping

- nos esquemas apresentados anteriormente, um processo permanece na memória principal até o fim de sua execução, inclusive nos momentos em que espera por um evento, como uma operação de leitura ou gravação;

- O *swapping* é uma técnica aplicada à gerência de memória para programas que esperam por memória livre para serem executados. O sistema escolhe um processo residente, que é transferido da memória principal para a memória secundária (*swap out*), geralmente disco;

- Posteriormente, o processo é carregado de volta da memória secundária para a memória principal (*swap in*) e pode continuar sua execução como se nada tivesse ocorrido.

Swapping

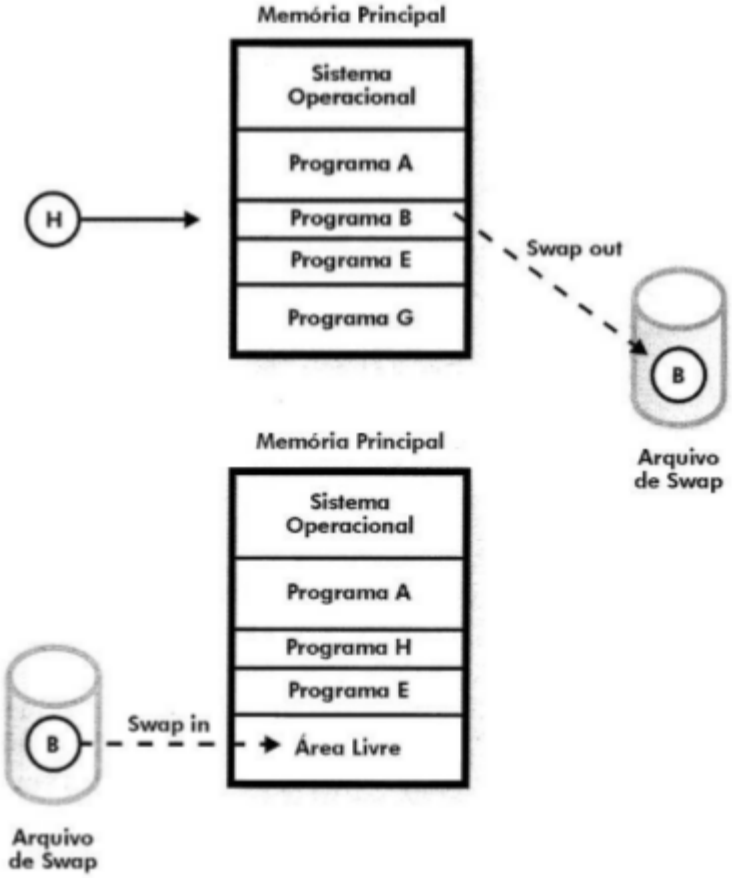


Figura 9.21: Swapping

Memória Virtual

- Técnica sofisticada e poderosa de gerência de memória, onde as memórias principal e secundária são combinadas, dando ao usuário a ilusão de existir uma memória muito maior que a capacidade real da memória principal;

- O conceito de memória virtual fundamenta-se em não vincular o endereçamento feito pelo programa aos endereços físicos da memória principal;

- Desta forma, programas e suas estruturas de dados deixam de estar limitados ao tamanho da memória física disponível, pois podem possuir endereços associados à memória secundária.

Sistema de Arquivos

- O armazenamento e a recuperação de informações é uma atividade essencial para qualquer tipo de aplicação;

- Um processo deve ser capaz de ler e gravar de forma permanente grande volume de dados em dispositivos como: discos, *pendrives*, fitas e etc., além de poder compartilhá-los com outros processos.

Arquivo

- Um arquivo é constituído por informações logicamente relacionadas. Estas informações podem representar instruções ou dados;
- Um arquivo executável contém instruções compreendidas pelo processador, enquanto um arquivo de dados pode ser estruturado livremente como um arquivo texto ou de forma mais rígida como um banco de dados relacional;
- Um arquivo é um conjunto de registros definidos pelo sistema de arquivos, tornando seu conceito abstrato e generalista.

Arquivo

- Um arquivo é identificado por um nome, composto por uma sequência de caracteres;
- Em alguns sistemas de arquivos é feita distinção entre caracteres alfabéticos maiúsculos e minúsculos;
- Regras como extensão máxima do nome e quais são os caracteres válidos também podem variar.

Arquivo

- Em alguns sistemas operacionais, a identificação de um arquivo é composta por duas partes separadas com um ponto. A parte após o ponto é denominada extensão do arquivo e tem como finalidade identificar o conteúdo do arquivo.

Extensão	Descrição
ARQUIVO. BAS	Arquivo fonte em BASIC
ARQUIVO.COB	Arquivo fonte em COBOL
ARQUIVO.EXE	Arquivo executável
ARQUIVO.OBJ	Arquivo objeto
ARQUIVO.TXT	Arquivo texto
ARQUIVO.PAS	Arquivo fonte em PASCAL

Arquivos - Atributos

- Cada arquivo possui informações de controle denominadas atributos;
- Os atributos variam dependendo do sistema de arquivos, porém alguns, como:
  - tamanho do arquivo,
  - proteção,
  - identificação do criador e
  - data de criação,

  estão presentes em quase todos os sistemas.

Diretórios (Pastas)

- A estrutura de diretórios é como o sistema organiza logicamente os diversos arquivos contidos em um disco;

- O diretório é uma estrutura de dados que contém entradas associadas aos arquivos, onde cada entrada armazena informações como localização física, nome, organização e demais atributos.

Estrutura de Diretórios em Árvore

Mais utilizada:

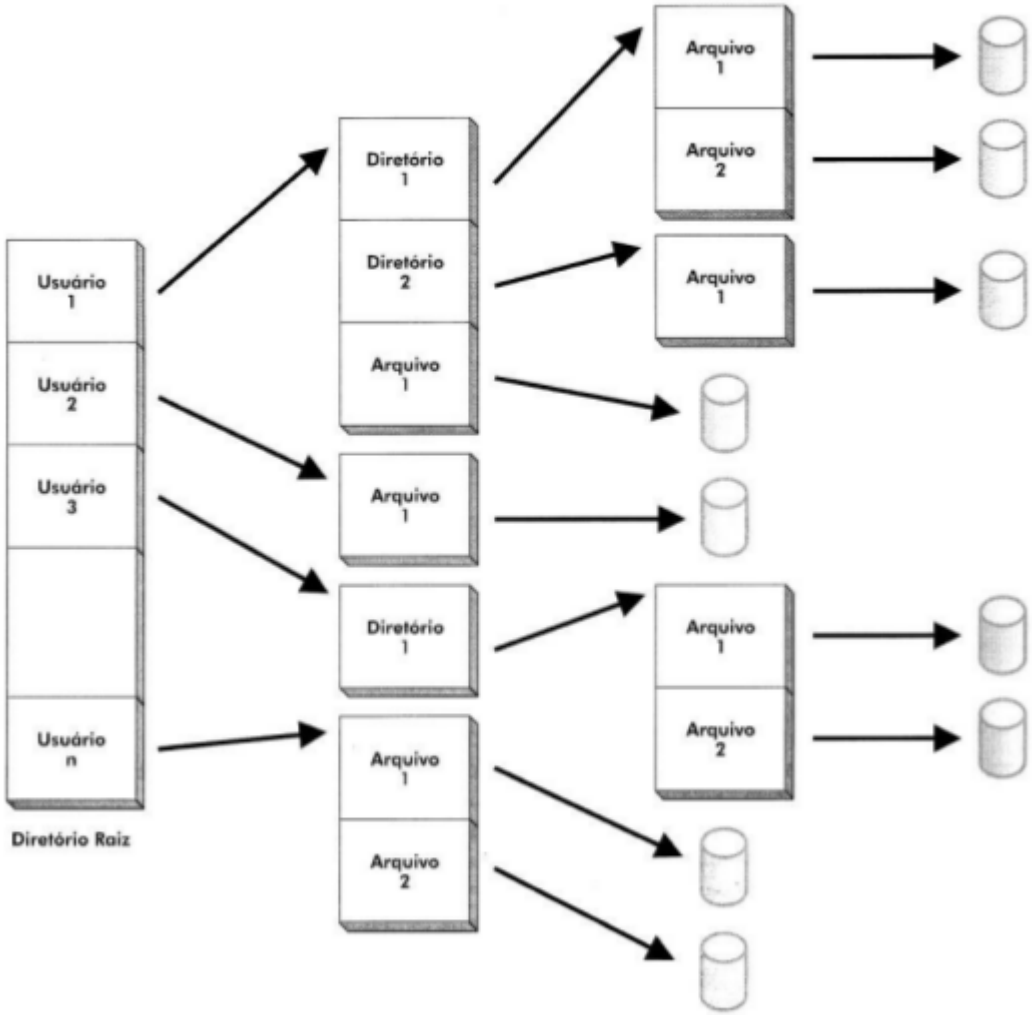


Figura 11.9: Estrutura de diretórios em árvore

Gerência de Espaço Livre em Disco

- A criação de arquivos em disco exige que o sistema operacional tenha o controle de quais áreas ou blocos no disco estão livres;

- Este controle é realizado utilizando-se uma estrutura de dados que armazena informações que possibilitam ao sistema de arquivos gerenciar o espaço livre do disco;

- Nesta estrutura é possível identificar blocos livres que poderão ser alocados a um novo arquivo.

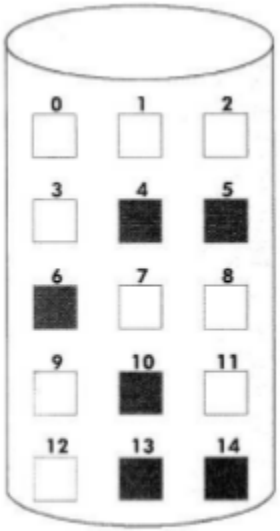
Gerência de Alocação de Espaço em Disco

- A **alocação contígua** consiste em armazenar um arquivo em blocos sequencialmente dispostos no disco;

- Neste tipo de alocação, o sistema localiza um arquivo através do endereço do primeiro bloco e da sua extensão em blocos.

Gerência de Alocação de Espaço em Disco

Alocação Contígua:



Arquivo	Bloco	Extensão
A. TXT	4	3
B. TXT	10	1
C. TXT	13	2

Figura 11.12: Alocação contígua

Gerência de Alocação de Espaço em Disco

- Na **alocação encadeada**, um arquivo pode ser organizado como um conjunto de blocos ligados logicamente no disco, independente da sua localização física;

- Cada bloco deve possuir um ponteiro para o bloco seguinte do arquivo e assim sucessivamente.

Gerência de Alocação de Espaço em Disco

Alocação Encadeada:

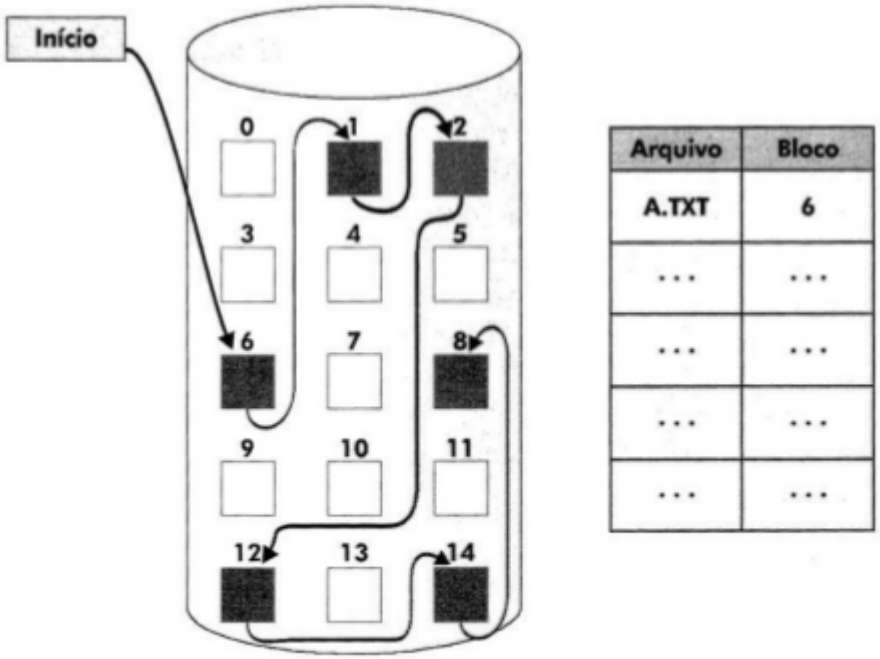


Figura 11.14: Alocação encadeada

Gerência de Alocação de Espaço em Disco

- A **alocação indexada** soluciona uma das principais limitações da alocação encadeada, que é a impossibilidade do acesso direto aos blocos dos arquivos;

- O princípio desta técnica é manter os ponteiros de todos os blocos do arquivo em uma única estrutura denominada bloco de índice.

Gerência de Alocação de Espaço em Disco

Alocação Indexada:

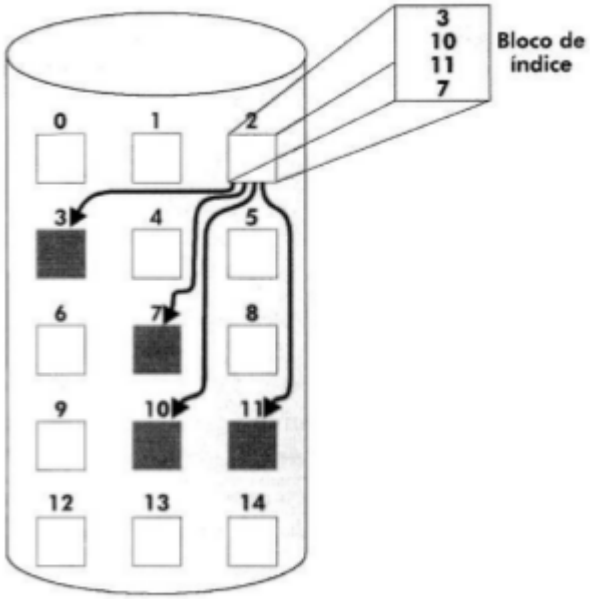


Figura 11.15: Alocação indexada

Gerência de Entrada e Saída

- Uma das principais funções dos sistemas operacionais é controlar todos os dispositivos de entrada/saída de um computador;
- O S.O. precisa emitir comandos para os dispositivos, capturar interrupções e manipular erros;
- Ele também fornece uma interface entre os dispositivos e o resto do sistema, interface esta que deve ser simples e fácil de usar. Tanto quanto possível, ela deve ser a mesma para todos os dispositivos, ou seja, deve independer do dispositivo.

Gerência de Entrada e Saída

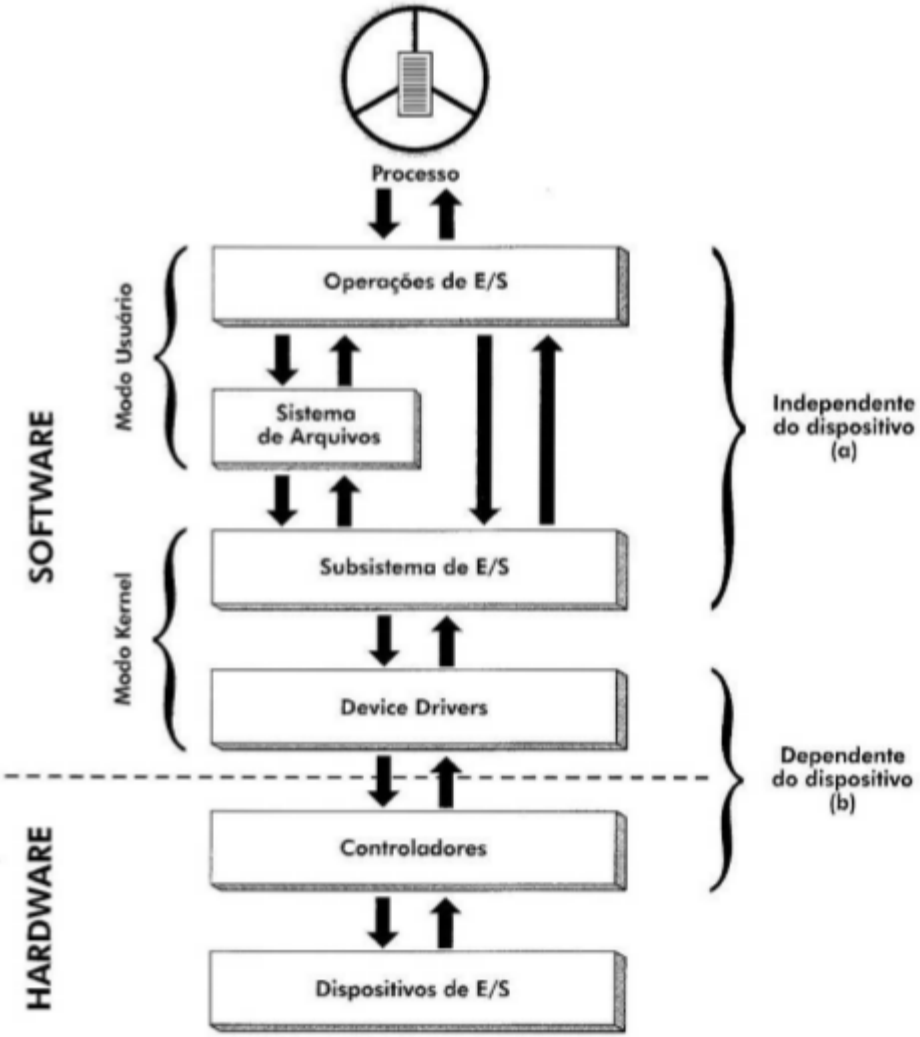


Figura 12.1: Arquitetura de camadas da gerência de dispositivos

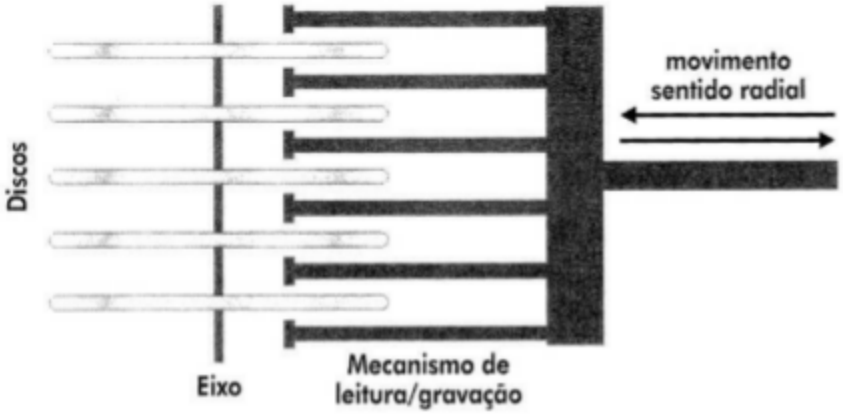
Discos

- Quase todos os computadores possuem discos para armazenamento de informação. Armazenar a informação em disco tem as seguintes vantagens sobre guardá-las na memória principal:
  - a capacidade de armazenamento disponível é muito grande;
  - o preço por bit armazenado é muito menor e
  - a informação não se perde quando se desliga a máquina.

Hardware de Disco

- Um disco magnético é constituído por vários discos sobrepostos, unidos por um mesmo eixo vertical, girando a uma velocidade constante;
- Cada disco é composto por trilhas concêntricas, que por sua vez são divididas em setores; apesar dos setores situados na parte mais externa do disco serem fisicamente maiores que os mais internos, o espaço extra não é utilizado;
- As trilhas dos diferentes discos que ocupam a mesma posição vertical formam um cilindro;
- Para a superfície de cada disco existe um mecanismo de leitura/gravação. Todos os mecanismos de leitura/gravação são conectados a um braço que se movimenta entre os vários cilindros dos discos no sentido radial.

Hardware de Disco



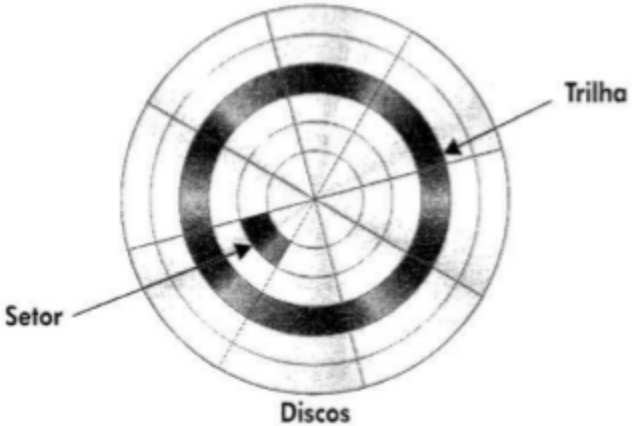


Figura 12.10: Estrutura de um disco magnético

Referências

- ERDEI, Jacob. **Basic Structure for a Pentium Microprocessor**. Disponível em: <https://www.pctechguide.com/cpu-architecture/basic-structure-of-a-pentium-microprocessor>. Acesso em: 13 jul. 2017.
- MACHADO, F. B.; MAIA, L. P. **Arquitetura de Sistemas Operacionais**. 3. ed. Rio de Janeiro: LTC, 2010.
- PANATTA, B.; SANTI, D. de; MOERSCHBACHER, G.; LIMA, L. G. de. **Esquema Geral de Funcionamento do Processador**. Disponível em: <http://sca.unioeste-foz.br/~habib/x/trabalhos/grupoa4/public_html/processador.html>. Acesso em: 14 jul. 2017.
- UNIVERSIDADE FEDERAL FLUMINENSE. **Organização de Computadores II - processadores**. Disponível em: <http://orgcomp2.ic.uff.br/processadores.php>. Acesso em: 14 jul. 2017.