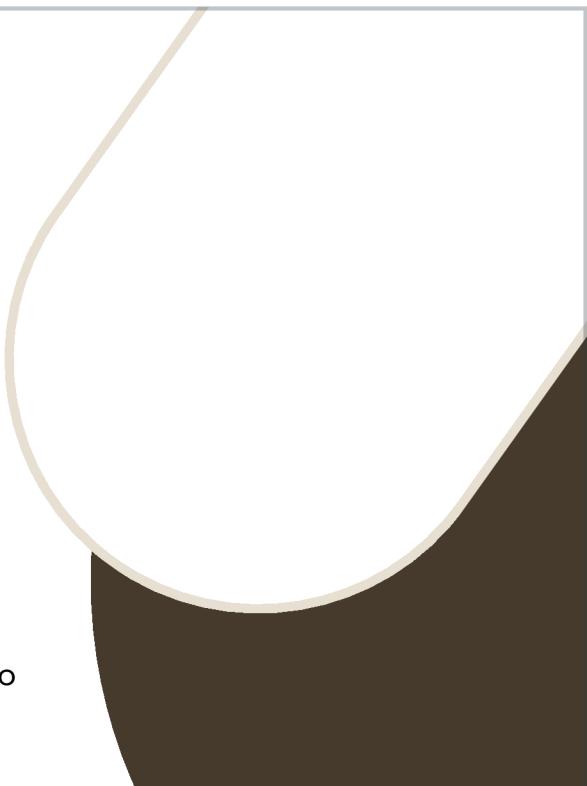


SISTEMAS DE ARQUIVO

1º Seminário de introdução à computação

Participantes: Matheus de oliveira e Daniel Pinheiro

Professora: Alessandra Hauck



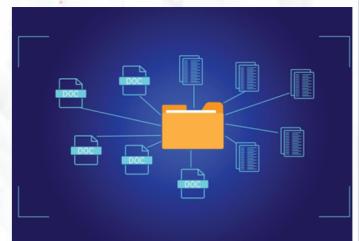
ARQUIVOS

1. CONCEITO DE ARQUIVOS

- Arquivos são unidades lógicas de informação criadas por processos.
- Cada arquivo é independente, mesmo que armazenado no mesmo disco.
- Arquivos funcionam como um espaço de endereçamento e não como memória RAM.
- Os arquivos são persistentes, ou seja, continuam existindo após o fim do processo que os criou.

2. SISTEMA DE ARQUIVOS

- Os arquivos são gerenciados pelo sistema operacional.
- São estruturados, nomeados, acessados, protegidos e implementados de acordo com o projeto do sistema operacional.
- A parte do sistema operacional responsável por isso é chamada de sistema de arquivos.



3. NOMEAÇÃO DE ARQUIVOS

- Arquivo é uma abstração para armazenar informações.
- O nome do arquivo é a principal forma de identificá-lo.
- Alguns sistemas (como UNIX) diferenciam maiúsculas de minúsculas nos nomes.
- Existem convenções de nomes e extensões (como .txt, .pdf, .jpg).

4. ESTRUTURA DE ARQUIVOS

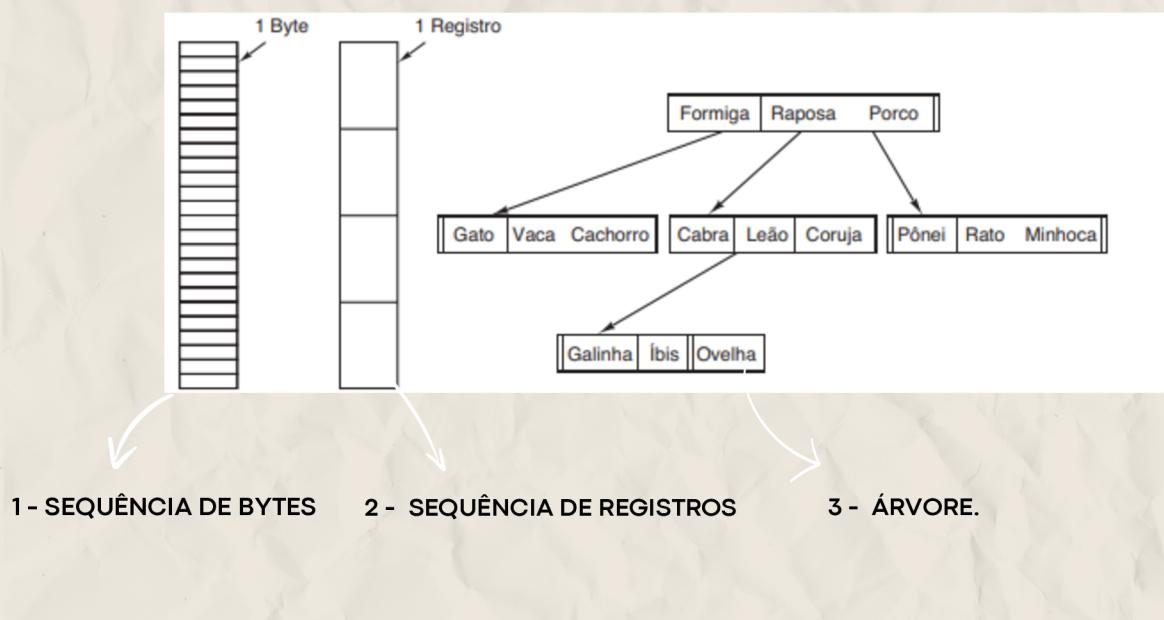
- Arquivos podem ser estruturados como:
- Sequência de bytes (mais comum).
- Sequência de registros (cada registro tem tamanho fixo).
- Árvore (com registros interligados por chaves de busca).

5. TIPOS DE ARQUIVOS

- Arquivos regulares: contêm dados e informações do usuário.
- Diretórios: armazenam referências para outros arquivos.
- Arquivos especiais de caracteres: usados por dispositivos como terminais e impressoras.
- Arquivos especiais de blocos: usados para modelar discos.

Estrutura de Arquivos

3 TIPOS DE ARQUIVOS



6. ATRIBUTOS DE ARQUIVOS

- Os arquivos possuem atributos (metadados), como:
- Nome, tamanho, data de criação/modificação.
- Flags (por exemplo: somente leitura, oculto, binário).
- Campos de controle como tamanho do registro e posição da chave.

7. OPERAÇÕES COM ARQUIVOS

- Operações comuns incluem:
- Create: cria um novo arquivo.
- Delete: remove um arquivo existente.
- Open: abre um arquivo para acesso.
- Close: fecha um arquivo.
- Read: lê dados do arquivo.
- Write: escreve dados no arquivo.
- Append: adiciona dados ao final do arquivo.
- Seek: reposiciona o ponteiro do arquivo.
- Get Attributes: obtém atributos do arquivo.
- Set Attributes: modifica atributos.
- Rename: renomeia o arquivo.

8 – Exemplo de Programa UNIX

- Um programa simples chamado copyfile copia o conteúdo de um arquivo para outro.
- Usa chamadas de sistema como open(), read(), write() e close().
- O código define buffer, modo de acesso e faz verificação de erros.

Exemplo

```
/* Programa que copia arquivos. Verificacao e relato de erros é minimo. */
#include <sys/types.h>                                /* inclui os arquivos de cabecalho necessarios */
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>

#define BUF_SIZE 4096
#define OUTPUT_MODE 0700

int main(int argc, char *argv[])
{
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];
    if (argc != 3) exit(1);                                /* erro de sintaxe se argc for 3 */

    /* Abre o arquivo de entrada e cria o arquivo de saida*/
    in_fd = open(argv[1], O_RDONLY);                      /* abre o arquivo de origem */
    if (in_fd < 0) exit(2);                                /* se nao puder ser aberto, saia */
    out_fd = creat(argv[2], OUTPUT_MODE);                  /* cria o arquivo de destino */
    if (out_fd < 0) exit(3);                                /* se nao puder ser criado, saia */

    /* Laco de copia*/
    while (TRUE) {
        rd_count = read(in_fd, buffer, BUF_SIZE);          /* le um bloco de dados */
        if (rd_count <= 0) break;                            /* se fim de arquivo ou erro, sai do laco */
        wt_count = write(out_fd, buffer, rd_count);          /* escreve dados */
        if (wt_count <= 0) exit(4);                            /* se wt_count <= 0 e um erro */
    }

    /* Fecha os arquivos*/
    close(in_fd);
    close(out_fd);
    if (rd_count == 0) exit(0);                            /* nenhum erro na ultima leitura */
    else exit(5);                                         /* erro na ultima leitura */
}
```

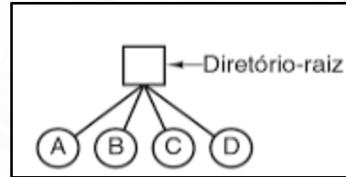
4.2 Diretórios

Para gerenciar os arquivos, os sistemas de arquivos geralmente possuem diretórios ou pastas, que em vários sistemas também são considerados arquivos. Nesta seção, abordaremos os diretórios, sua estrutura, características e funções.

4.2.1 Sistemas de diretório em nível único

O sistema de diretório mais simples consiste em um único diretório, chamado de diretório principal, que contém todos os arquivos. Esse modelo era comum em primeiros computadores pessoais e no supercomputador CDC 6600, usado por múltiplos usuários, devido à sua simplicidade e facilidade de localizar arquivos. Ainda é utilizado em sistemas embarcados, como telefones, câmeras digitais e players de música.

EXEMPLO DE UM SISTEMA DE NÍVEL ÚNICO



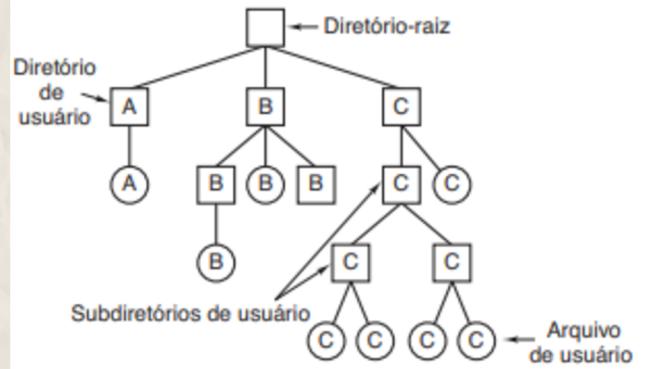
DIRETÓRIO RAIZ

É O NÍVEL MAIS ALTO DA HIERARQUIA E SERVE COMO PONTO DE PARTIDA PARA TODA A ESTRUTURA.

4.2.2 Sistemas de diretórios hierárquicos

O SISTEMA DE NÍVEL ÚNICO, EMBORA ADEQUADO PARA APLICAÇÕES SIMPLES E USADO NOS PRIMEIROS COMPUTADORES, TORNA-SE IMPRATICÁVEL PARA USUÁRIOS MODERNOS COM MILHÕES DE ARQUIVOS. POR ISSO, É NECESSÁRIA UMA ESTRUTURA HIERÁRQUICA DE DIRETÓRIOS, ORGANIZADA EM FORMA DE ÁRVORE, PERMITINDO AOS USUÁRIOS CRIAR DIRETÓRIOS PERSONALIZADOS. EM REDES DE EMPRESAS, CADA USUÁRIO PODE TER UM DIRETÓRIO-RAIZ, FACILITANDO A CRIAÇÃO DE SUBDIRETÓRIOS PARA PROJETOS. ESSA ESTRUTURA É ADOTADA PELA MAIORIA DOS SISTEMAS DE ARQUIVOS MODERNOS DEVIDO À SUA EFICIÊNCIA NA ORGANIZAÇÃO.

EXEMPLO DE UM SISTEMA HIERÁRQUICO DE DIRETÓRIOS



4.2.3 Nomes de caminhos

CAMINHOS ABSOLUTOS:

Nomes de caminhos absolutos sempre inicializam no diretório-raiz e são únicos. Os caminhos absolutos não dependem de onde são usados, sendo embasado na raiz do sistema de arquivos. Outra coisa importante é que, não importa qual caractere é usado: se o primeiro caractere do nome do caminho for o separador, então o caminho será absoluto.

SEPARADORES:

- NO UNIX, OS COMPONENTES DO CAMINHO SÃO SEPARADOS POR /.
- NO WINDOWS, O SEPARADOR É \.
- NO MULTICS, ERA >.

EXEMPLOS DE SEPARADORES:

WINDOWS	\USR\AST\CAIXAPOSTAL
MULTICS	>USR>AST>CAIXAPOSTAL
UNIX	/USR/AST/CAIXAPOSTAL

CAMINHOS RELATIVOS:

diferente do absoluto, ele depende de onde ele é usado (precisa ser combinado com outro caminho). Usado para executar comandos mais curtos, inclui símbolos sendo os principais ("." e ".."). É bom lembrar que todos os nomes de caminho que não começem no diretório-raiz são assumidos como relativos ao diretório de trabalho.

ENTRADAS ESPECIAIS :

“.” PONTO: REFERE-SE AO DIRETÓRIO ATUAL

“..” PONTOPONTO: REFERE-SE A SEU PAI — OU SEJA, O DIRETÓRIO “ACIMA” DO ATUAL NA HIERARQUIA.

Métodos de alocação de arquivos

1- Alocação contígua:

Esse é o método mais simples de alocação de arquivos, no qual cada arquivo é guardado no disco como um bloco contínuo de dados. Suponha que um disco possua blocos de 1 KB. Um arquivo pequeno de 10KB seria armazenado em 10 blocos seguidos.



VANTAGENS:

- Possui uma fácil implementação (o controle da localização de cada arquivo no disco é feito por meio de um único número controle da localização de cada arquivo no disco é feito por meio de um único número) o arquivo pode ser lido do disco de uma única vez

DESVANTAGENS:

- Esse tipo de alocação pode causar grande perda de espaço útil. Além disso essa estratégia só é viável quando o tamanho máximo do arquivo é conhecido no momento de sua criação (pois é necessário saber previamente o tamanho total do arquivo ou a quantidade de blocos que ele irá ocupar)

Arquivo na memória



Métodos de alocação de arquivos

2 - Alocação com Lista Ligada:

Nesse tipo de alocação, utiliza-se uma lista encadeada para indicar os blocos ocupados no disco pelo arquivo. Dessa forma, não é mais necessário que os arquivos ficam armazenados em blocos consecutivos no disco. Além disso, a primeira palavra de cada bloco funciona como um ponteiro para o bloco seguinte, enquanto o restante do espaço é utilizado para guardar os dados do arquivo.

VANTAGENS:

- Evita desperdício de espaço devido à fragmentação externa
- A entrada no diretório precisa apenas guardar o endereço do primeiro bloco, já que cada bloco aponta para o próximo.
- Blocos podem ser alocados livremente, o que possibilita que os arquivos aumentem de tamanho enquanto houver espaço disponível

DESVANTAGENS:

- Possui uma implementação bem mais complicada devido a grande ramificação de arquivos desse método.

 Arquivo na memória



Métodos de alocação de arquivos

3 – Alocação com Lista Usando Tabela:

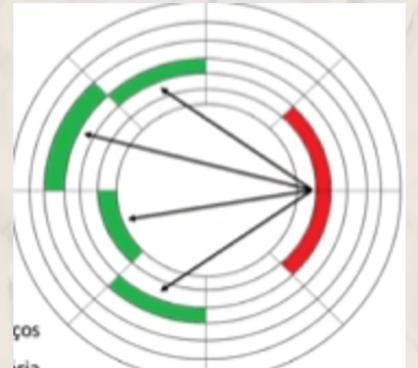
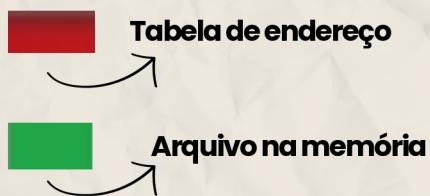
Nesse método vamos ter uma tabela de endereços dentro do disco que armazena os ponteiros para cada bloco do arquivo.

VANTAGENS:

- O acesso aleatório aos dados se torna mais simples, pois todos os ponteiros ficam reunidos em um único local na memória secundária.

DESVANTAGENS:

- Há um consumo de memória adicional para manter a tabela com as informações.



Métodos de alocação de arquivos

4 - Alocação com Lista Usando um Índice:

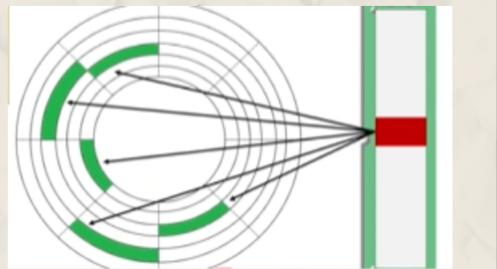
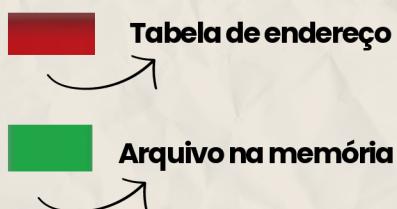
Contém também a tabela de endereço no disco, porém agora ela está alocada na memória ram.

VANTAGENS:

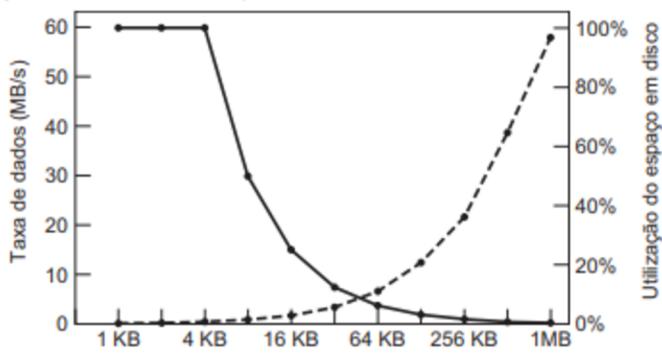
- Como essa tabela está na memória principal, ela pode ser consultada diretamente, sem a necessidade de acessar o disco.

DESVANTAGENS:

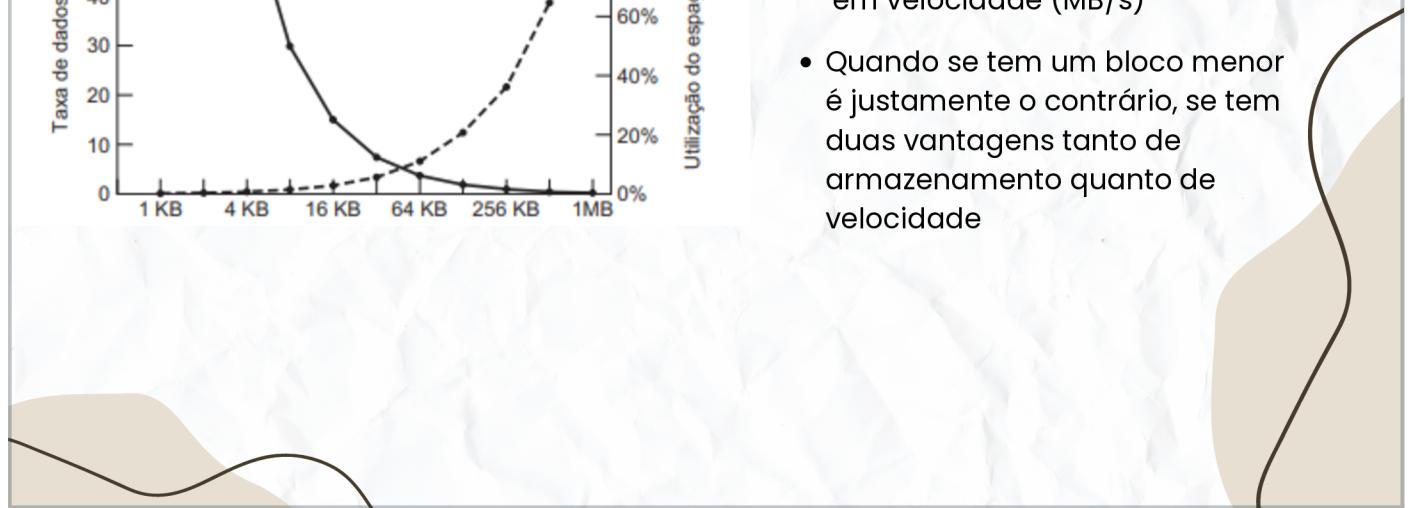
- A tabela precisa permanecer carregada na memória o tempo inteiro, o que resulta em consumo adicional de espaço de memória.



Eficiência do espaço em disco



- Quando se tem um bloco maior, se perde a eficiência de espaço de armazenamento e também em velocidade (MB/s)
- Quando se tem um bloco menor é justamente o contrário, se tem duas vantagens tanto de armazenamento quanto de velocidade



Backups (cópias de segurança) do sistema de arquivos

perda de um sistema de arquivos geralmente é um desastre muito mais grave do que a perda do próprio computador. Se o computador for danificado por fogo, uma descarga elétrica ou até mesmo por café derramado no teclado, isso pode ser frustrante e gerar custos, mas ainda assim é menos crítico.

A melhor solução para esse tipo de problema é fazer cópias de segurança, os famosos backups. Assim garantindo que seus dados não sejam totalmente perdidos após algum tipo de acidente ou falha do sistema.



Backups

CÓPIA FÍSICA:

- inicia no bloco 0 do disco e grava todos os blocos, em ordem, no disco de destino, até que o último seja copiado. Esse tipo de programa é tão simples que pode ser desenvolvido com total ausência de erros — algo raro quando se trata de programas úteis.

CÓPIA LÓGICA:

- O processo inicia em um ou mais diretórios definidos e, de forma recursiva, copia todos os arquivos e subdiretórios que tenham sido modificados desde uma data de referência — como o último backup, no caso de uma cópia incremental, ou a instalação do sistema, no caso de uma cópia completa.

• uma cópia incremental ele só faz backups de arquivos que foram modificados

Diretório que não foi alterado → Diretório que foi alterado → Arquivo que não foi alterado

(a) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

(b) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

(c) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

(d) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

Uma leitura é feita sobre os diretórios e arquivos modificados e colocados em forma de tabela.

- **Obs:** na primeira etapa todos os diretórios são assinalados como modificados.
- **Obs:** na segunda etapa os diretórios são assinalados como modificados para marcar o caminho.
- **Obs:** na terceira etapa é parecida com a segunda fase porém sendo assinaladas apenas os diretórios.
- **Obs:** na quarta etapa apenas os arquivos modificados são assinalados.

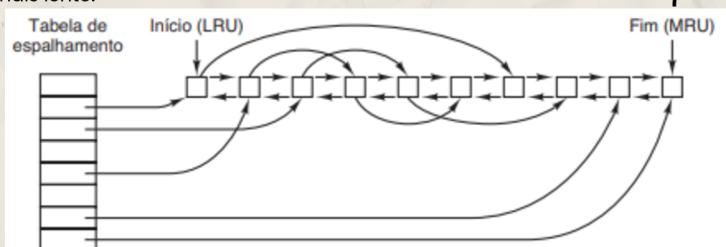
Desempenho do sistema de arquivos

O acesso à memória é muito mais rápido que acesso ao disco

- Tempo para ler uma palavra da memória: 10 ns
- Tempo para ler um bloco do disco: 4x mais lento (pelo menos)

Cache de blocos

Quando queremos otimizar o desempenho de algo que é mais lento usamos um cache para armazenar informações muito utilizadas, assim evitando uma repetição de leitura no dispositivo mais lento.



- Lista única permite LRU puro
- LRU puro pode levar a inconsistências
Blocos modificados que não foram sincronizados no disco
- LRU modificado
Blocos modificados com metadados devem ser logo escritos no disco.
- Como evitar perder dados do usuário?

Chamada de sistema **sync**
Chama o **sync** a cada 30 segundos (daemon)
sync ele pega todos os blocos que foram modificados da cache de blocos e copia pro DISCO

Todas as vezes que um bloco precisar ser lido, antes de ele ser lido no disco o sistema operacional vai buscar ler esse arquivo na cache

Desfragmentação de disco



- Durante a instalação inicial do sistema operacional, os programas e arquivos necessários são gravados de forma sequencial, iniciando no começo do disco, com cada um sendo colocado logo após o outro.
- O desempenho pode ser recuperado reorganizando os arquivos para que fiquem armazenados de forma contígua e concentrando todo (ou a maior parte) do espaço livre em uma ou mais áreas contínuas do disco.
- A desfragmentação é mais eficaz em sistemas de arquivos que possuem uma quantidade significativa de espaço livre localizado de forma contínua próximo ao final da partição.

OBRIGADO

