O conceito de arquivo

PROFESSOR: Jheymesson A. Cavalcanti

DISCIPLINA: SISTEMAS OPERACIONAIS II

1. O sistema de arquivos

Em computação, existe a necessidade de armazenar informações para uso posterior, como programas e dados;

Atualmente, parte importante do uso de um computador consiste em recuperar e apresentar informações previamente armazenadas, como:

- Documentos;
- Fotografias;
- Músicas;
- Vídeos;
- Etc.

O próprio sistema operacional também precisa manter informações armazenadas para uso posterior, como programas, bibliotecas e configurações.

1. O sistema de arquivos

Para simplificar o armazenamento e busca dessas informações, surgiu o conceito de **ARQUIVO**.

Mas o que seria um arquivo?

1. O sistema de arquivos

Um arquivo é essencialmente uma sequência de bytes armazenada em um dispositivo físico não volátil, como um disco rígido ou de estado sólido.

Também pode ser visto, pelo lado do usuário, como a unidade básica de armazenamento de dados;

Cada arquivo possui um nome, ou outra referência, que permite sua localização e acesso;

Arquivos são extremamente versáteis em conteúdo e capacidade, podendo conter desde um texto com alguns poucos caracteres até vídeos com dezenas de gigabytes.

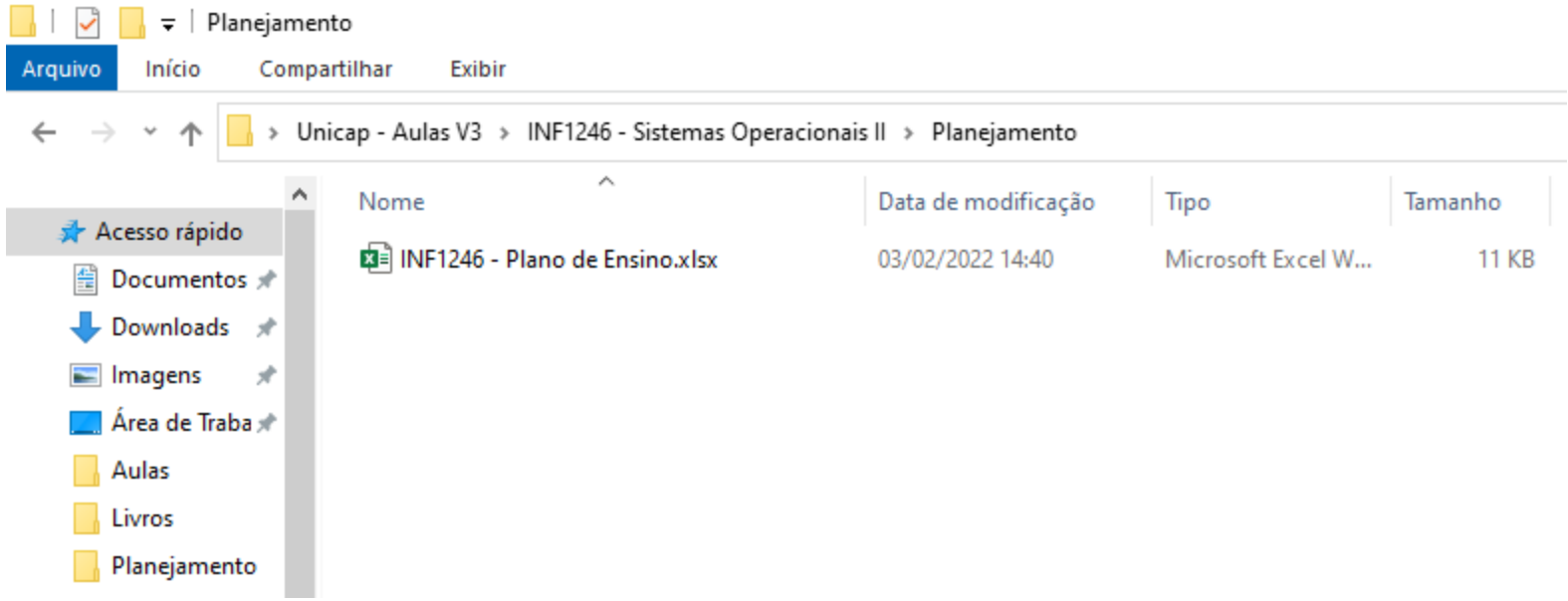
1. O sistema de arquivos

Exemplos de tipos de arquivos:

1. O sistema de arquivos

Como um dispositivo de armazenamento pode conter milhões de arquivos, estes são organizados em estruturas hierárquicas denominadas **DIRETÓRIOS**;

Os diretórios facilitam a localização e acesso de arquivos pelos usuários.



1. O sistema de arquivos

A organização do conteúdo dos arquivos e diretórios dentro de um dispositivo físico é denominada **SISTEMA DE ARQUIVOS**;

Ele pode ser visto como uma imensa estrutura de dados armazenada de forma persistente no dispositivo físico;

Existe um grande número de sistemas de arquivos, como:

- NTFS (nos sistemas Windows);
- Ext2/Ext3/Ext4 (Linux);
- HPFS (MacOS);
- FFS (Solaris);
- FAT (usado em pendrives USB, câmeras fotográficas digitais e leitores MP3).

1. O sistema de arquivos

Além disso, um dispositivo físico é estruturado em um ou mais volumes (ou partições);

Cada volume pode armazenar um sistema de arquivos próprio;

Exemplo:

- Um mesmo disco pode conter volumes com diferentes sistemas de arquivos, como FAT, NTFS ou EXT4, por exemplo.

2. Atributos e operações

Um arquivo é uma unidade de armazenamento de informações que podem ser documentos, imagens, código executável, etc. (**CONTEÚDO**);

Além do conteúdo, um arquivo é caracterizado por ATRIBUTOS ou METADADOS e por OPERAÇÕES;

Atributos são informações adicionais relativas ao conteúdo.

Operações são ações que podem ser realizadas sobre o conteúdo e/ou sobre os atributos do arquivo.

2. Atributos e operações

Os atributos dos arquivos variam de acordo com o sistema de arquivos utilizado;

Os atributos mais usuais, presentes na maioria dos sistemas, são:

Nome: string que identifica o arquivo para o usuário, como: “foto1.jpg”, “hello.c”, etc.

Tipo: indicação do formato dos dados contidos no arquivo, como áudio, vídeo, imagem, texto, etc.

Tamanho: indicação do tamanho do conteúdo do arquivo, geralmente em bytes.

2. Atributos e operações

Localização: indicação do dispositivo físico onde o arquivo se encontra e da posição do arquivo dentro do mesmo.

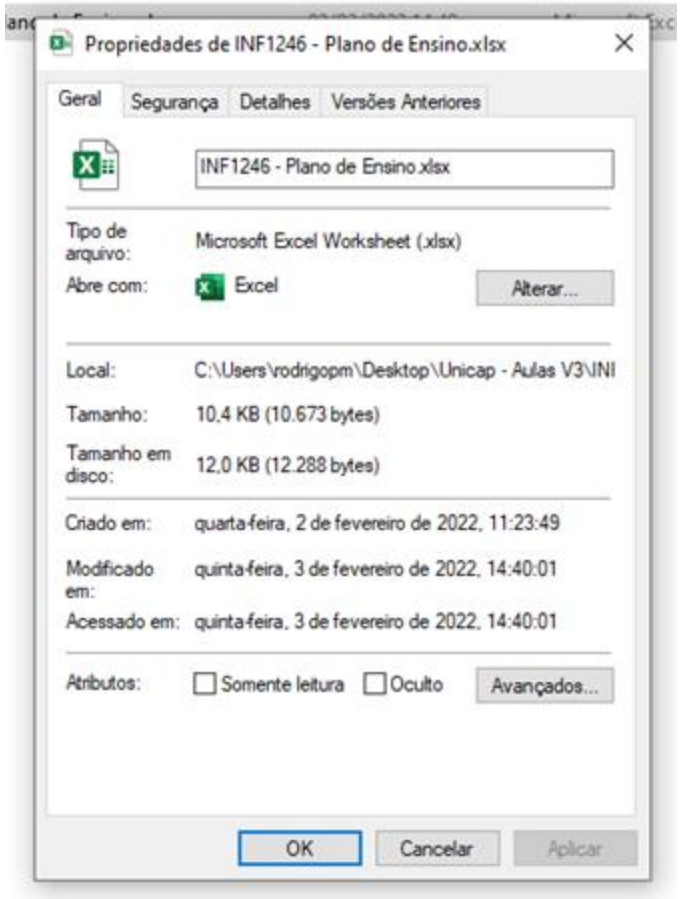
Datas: para fins de gerência, é importante manter as datas mais importantes relacionadas ao arquivo, como suas datas de criação, de último acesso e de última modificação do conteúdo.

Proprietário: em sistemas multiusuários, cada arquivo tem um proprietário, que deve estar corretamente identificado.

Permissões de acesso: indicam que usuários têm acesso àquele arquivo e que formas de acesso são permitidas (leitura, escrita, remoção, etc.).

Etc.

2. Atributos e operações



2. Atributos e operações

As aplicações e o sistema operacional usam arquivos para armazenar e recuperar dados;

O acesso aos arquivos é feito através de um conjunto de operações;

Tais operações geralmente são implementadas sob a forma de chamadas de sistema e funções de bibliotecas;

As operações básicas envolvendo arquivos são:

Criar: a criação de um novo arquivo implica em alocar espaço para ele no dispositivo de armazenamento e definir valores para seus atributos.

Escrever: permite transferir dados na memória da aplicação para o arquivo no dispositivo físico. Os novos dados podem ser adicionados ao final do arquivo ou sobrescrever dados já existentes.

Ler: permite transferir dados presentes no arquivo para uma área de memória da aplicação;

2. Atributos e operações

Fechar: ao concluir o uso do arquivo, a aplicação deve informar ao sistema operacional que o mesmo não é mais necessário, a fim de liberar as estruturas de gerência do arquivo mantidas na memória do núcleo;

Remover: para eliminar o arquivo do dispositivo, descartando seus dados e liberando o espaço ocupado por ele.

Alterar atributos: para modificar os valores dos atributos do arquivo, como nome, proprietário, permissões, datas, etc.

Abrir: antes que uma aplicação possa ler ou escrever dados em um arquivo, ela deve solicitar ao sistema operacional a “abertura” desse arquivo. O sistema irá então verificar se o arquivo desejado existe, verificar se as permissões associadas ao arquivo permitem aquele acesso, localizar seu conteúdo no dispositivo de armazenamento e criar uma referência para ele na memória da aplicação.

Etc.

3. Formatos de arquivos

Arquivos permitem armazenar dados para uso posterior;

A forma como esses dados são estocados dentro do arquivo é denominada **FORMATO DE ARQUIVO**;

Alguns exemplos de formatos de arquivo:

- Sequência de bytes;
- Arquivos de registros;
- Arquivos de texto;
- Arquivos de código;
- Etc.

3. Formatos de arquivos

1. Sequência de bytes:

Em sua forma mais simples, **TODO** arquivo contém basicamente uma sequência de bytes;

Essa sequência de bytes pode ser estruturada de forma a representar diferentes tipos de informação, como:

- Imagens;
- Música;
- Textos;
- Código executável;
- Etc.

A estrutura interna do arquivo pode ser entendida pelo núcleo do sistema operacional ou somente pelas aplicações que acessam esse conteúdo.

3. Formatos de arquivos

1. Sequência de bytes:

Obs:

- O núcleo do sistema geralmente reconhece apenas alguns poucos formatos de arquivos, como binários executáveis e bibliotecas;
- Os demais formatos de arquivos são vistos pelo núcleo apenas como sequências de bytes opacas, sem um significado específico;
- Para estes formatos, cabe às aplicações interessadas interpretá-las.

3. Formatos de arquivos

2. Arquivos de registros:

Alguns núcleos de sistemas operacionais oferecem arquivos com estruturas internas que vão além da simples sequência de bytes;

Exemplo:

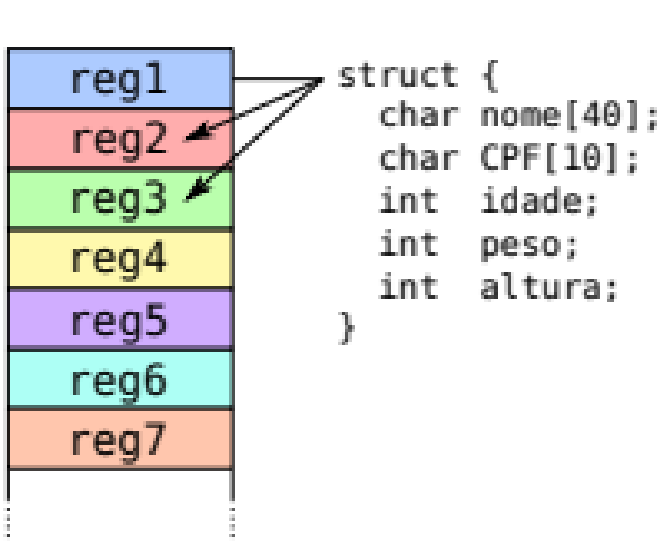
- O sistema OpenVMS proporciona arquivos baseados em registros;
- Seu conteúdo é visto pelas aplicações como uma sequência linear de registros de tamanho fixo ou variável, e também arquivos indexados;
- Nestes arquivos indexados podem ser armazenados pares {chave/valor}, de forma similar a um banco de dados relacional.

Obs:

- Em alguns sistemas operacionais onde estes arquivos não são suportados pelo núcleo, o suporte pode vir de bibliotecas e algumas linguagens de programação;
- Exemplos: bibliotecas Berkeley DB e SQLite, disponíveis em plataformas UNIX.

3. Formatos de arquivos

2. Arquivos de registros:



nome (chave)	telefone (valor)
daniel	9977-1173
marina	9876-5432
henrique	8781-9750
gabriel	8858-8286
renata	9663-9293
andressa	8779-5538
guilherme	9979-4166

3. Formatos de arquivos

3. Arquivos de textos:

Um tipo de arquivo de uso muito frequente é o arquivo de texto puro (plain text);

Esse tipo de arquivo é usado para armazenar informações textuais simples, como:

- Códigos-fonte de programas;
- Arquivos de configuração;
- Páginas HTML;
- Dados em XML;
- Etc.

Um arquivo de texto é formado por linhas de caracteres de tamanho variável, separadas por caracteres de controle;

Exemplo:

- Nos sistemas UNIX, as linhas são separadas por um caractere New Line (ASCII 10 ou “\n”).

3. Formatos de arquivos

3. Arquivos de textos:

Exemplo:

- O arquivo de texto .c seria armazenado usando a seguinte sequência de bytes em um sistema UNIX:

```
1 int␣main()
2 {
3 ␣␣printf("Hello,␣world\n");
4 ␣␣exit(0);
5 }
```

```
1 0000  69 6e 74 20 6d 61 69 6e 28 29 0a 7b 0a 20 20 70
2        i  n  t  ␣  m  a  i  n  (  ) \n  {  \n ␣  ␣  p
3 0010  72 69 6e 74 66 28 22 48 65 6c 6c 6f 2c 20 77 6f
4        r  i  n  t  f  (  "  H  e  l  l  o  ,  ␣  w  o
5 0020  72 6c 64 5c 6e 22 29 3b 0a 20 20 65 78 69 74 28
6        r  l  d  \  n  "  )  ;  \n ␣  ␣  e  x  i  t  (
7 0030  30 29 3b 0a 7d 0a
8        0  )  ;  \n  }  \n
```

3. Formatos de arquivos

4. Arquivos de código:

É usado em programas executáveis e bibliotecas compiláveis;

É dividido internamente em várias seções, para conter: código, tabelas de símbolos (variáveis e funções), listas de dependências (bibliotecas necessárias) e outras informações de configuração;

A organização interna de um arquivo de código depende do sistema operacional para o qual foi definido.

3. Formatos de arquivos

4. Arquivos de código:

Os formatos de código mais usuais atualmente são:

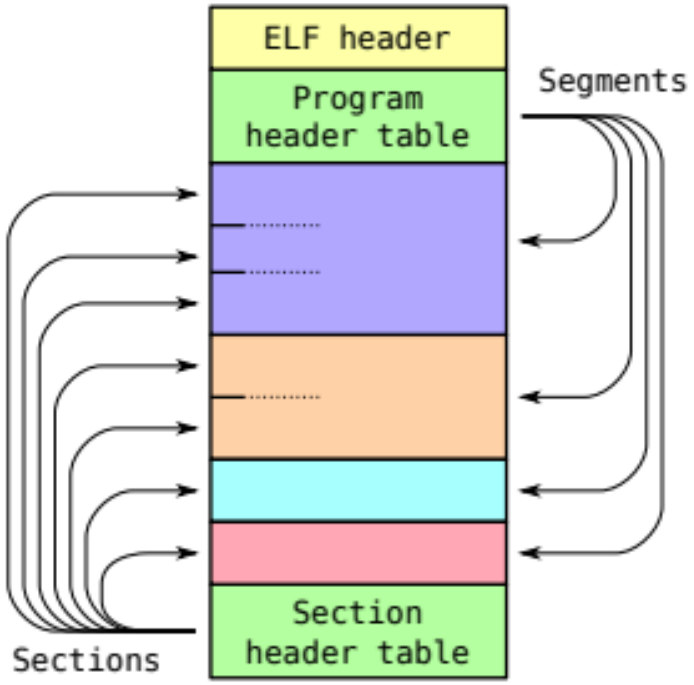
- **ELF (Executable and Linking Format):**
- Formato de arquivo usado para programas executáveis e bibliotecas na maior parte das plataformas UNIX modernas;
- É composto por um cabeçalho e várias seções de dados, contendo código executável, tabelas de símbolos e informações sobre relocação de código, usadas quando o código executável é carregado na memória.
- **PE (Portable Executable):**
- É o formato usado para executáveis e bibliotecas na plataforma Windows;
- Consiste basicamente em uma extensão do formato COFF (Common Object File Format), usado em plataformas UNIX mais antigas.

3. Formatos de arquivos

4. Arquivos de código:

Exemplo:

- Formato ELF:



4. Identificação de conteúdo

Um problema importante relacionado aos formatos de arquivos é a correta identificação de seu conteúdo pelos usuários e aplicações;

Já que um arquivo de dados pode ser visto como uma simples sequência de bytes, como é possível reconhecer que tipo de informação essa sequência representa?

Uma solução simples para esse problema consiste em usar parte do nome do arquivo para indicar o tipo do conteúdo;

Exemplos:

- Um arquivo “praia.jpg” contém uma imagem em formato JPEG;
- Um arquivo “entrevista.mp3” contém áudio em formato MP3.

A estratégia de extensão do nome é utilizada ainda hoje na maioria dos sistemas operacionais.

4. Identificação de conteúdo

Outra abordagem, frequentemente usada em sistemas UNIX, é usar alguns bytes no início do conteúdo do arquivo para a definição de seu tipo;

Esses bytes iniciais do conteúdo são denominados números mágicos (magic numbers).

Tipo de arquivo	bytes iniciais	Tipo de arquivo	bytes iniciais
Documento PostScript	`%!`	Documento PDF	`%PDF`
Imagem GIF	`GIF89a`	Imagem JPEG	`0xFF D8 FF`
Música MIDI	`MThd`	Classes Java	`0xCA FE BA BE`
Arquivo ZIP	`0x50 4B 03 04`	Documento RTF	`{\rtf1`

4. Identificação de conteúdo

Recentemente, a necessidade de transferir arquivos através de e-mail e de páginas Web levou à definição de um novo padrão de tipagem de arquivos;

O nome deste padrão é MIME (Multipurpose Internet Mail Extensions);

O padrão MIME define tipos de arquivos através de uma notação uniformizada na forma “tipo/subtipo”;

O padrão MIME é usado para identificar arquivos transferidos como anexos de e-mail e conteúdos obtidos de páginas Web.

4. Identificação de conteúdo

Tipo MIME	Significado
`application/java-archive`	Arquivo de classes Java
`application/msword`	Documento do Microsoft Word
`application/vnd.oasis.opendocument.text`	Documento do OpenOffice
`audio/midi`	Áudio em formato MIDI
`audio/mpeg`	Áudio em formato MP3
`image/jpeg`	Imagem em formato JPEG
`image/png`	Imagem em formato PNG
`text/csv`	Texto em formato CSV (*Comma-separated Values*)
`text/html`	Texto HTML
`text/plain`	Texto puro
`text/rtf`	Texto em formato RTF (*Rich Text Format*)
`text/x-csrc`	Código-fonte em C
`video/quicktime`	Vídeo no formato *Quicktime*

4. Identificação de conteúdo

OBS: Arquivos especiais

Além do armazenamento de dados do sistema operacional e de aplicações, o conceito de arquivo também pode ser usado como:

Abstração de dispositivos de baixo nível:

- /dev/ttyS0: porta de comunicação serial;
- /dev/sda1: partição de disco.

Abstração de interfaces do núcleo:

- /proc/cpuinfo: informações sobre processadores;
- /proc/3754/maps: mapa de memória do processo 3754

Canais de comunicação: sockets de rede, pipes;

Abstrações diversas:

- /dev/random: fonte de bytes aleatórios;
- /dev/null: “buraco negro” de dados.

O conceito de arquivo - Obrigado!

PROFESSOR: Jheymesson A. Cavalcanti

DISCIPLINA: SISTEMAS OPERACIONAIS II