Trabalhando com Arquivos em Linguagem C

Até o momento...

- As estruturas de dados abordadas armazenam as informações na **memória principal** do computador

- **Isso nem sempre é conveniente**

Até o momento...

- **Problemas:**
  - A informação é perdida
  - As estruturas de dados são limitadas, pois existe uma **quantidade** de informação que pode ser armazenada para resolver o problema

Algumas Informações...

- **Precisam ser mantidas para posterior uso:**
  - Cadastro de Alunos
  - Cadastro de Vendas de uma empresa
  - Dados de Pacientes
  - ...

Arquivo

- É armazenado em um dispositivo de **memória secundária**
- Pode ser lido e escrito por um programa
- Em programação, existem vários tipos de informações que podem ser armazenadas em arquivo

Arquivo é uma...

- **Sequência de bytes** que **reside** em um **dispositivo de armazenamento durável**, como:
  - Disco Rígido
  - CD
  - Pendrive
  - SSD
  - ...

Arquivo pode armazenar...

- Textos
- Imagens
- Áudios
- Vídeos
- ...

A extensão de um Arquivo...

- Determina qual o seu tipo (seu conteúdo):
  - .TXT
  - .PNG, .JPG
  - .MP3, .WAV, .FLAC
  - .AVI, .MPEG, .MKV

Um Arquivo possui...

- Atributos chamados **metadados**
- Trais metadados são utilizados pelo sistema de arquivos:
  - Tamanho (em bytes)
  - Usuário que o criou
  - Datas (criação, ultima modificação, …)
  - Permissões

Arquivos em Linguagem C

- Um arquivo é representado como uma **stream**

  Representação lógica (em memória) de um arquivo ou dispositivo de E/S:

  - Teclado
  - Monitor
  - ...

Arquivos em Linguagem C

- Em C, existem **DOIS tipos de streams:**
  - TEXTO
  - BINÁRIO

Arquivos em Linguagem C

- **TEXTO:**

  Sequência de linhas, onde cada linha contém zero ou mais caracteres e termina com um **\n**

Arquivos em Linguagem C

- **BINÁRIO:**

  Sequência de bytes que são lidos e gravados exatamente como estão armazenados em memória

Arquivos em Linguagem C

- **Streams padrão:**
  - **stdin**
  - **stdout**
  - **stderr**

Arquivos em Linguagem C

- **stdin:**

  Entrada de dados padrão de um programa
- **stdout:**

  Saída de dados padrão de um programa
- **stderr:**

  Saída de erros padrão de um programa

Arquivos em Linguagem C

- Em C, o **tipo de dados** que **implementa uma stream** é o **FILE**
- **FILE** é uma **struct** definida na biblioteca **<stdio.h>**

Struct FILE

```
typedef struct {
unsigned char *_p;  /*current position in (some) buffer*/
int _r;   /*read space left for reading*/
int _w;  /*write space left for writing*/
short _flags;        /*flags, below; this FILE is free if 0*/
short _file;    /*fileno, if Unix descriptor, else -1*/
} FILE;
```

Arquivos em Linguagem C

- Os nomes de arquivos são armazenados em **strings**

• Cada sistema operacional utiliza uma forma de indicar o caminho do arquivo

Arquivos em Linguagem C

- **Windows**

  char *filename = “c:\\Users\\Lucas\\arquivo.txt”;

- **Linux**

  char *filename = “/home/Lucas/arquivo.txt”;

Funções em C (stdio.h)

- **fopen()** = abre um arquivo
- **fclose()** = fecha um arquivo
- **ferror()** = retorna verdadeiro se ocorreu um erro
- **fputc()** = escreve um caracter em um arquivo
- **fgetc()** = lê um caracter de um arquivo

Funções em C (stdio.h)

- **fputs()** = escreve uma string em um arquivo
- **fgets()** = lê uma string de um arquivo
- **fwrite()** = escreve uma estrutura (struct) em um arquivo
- **fread()** = lê uma estrutura (struct) de um arquivo
- **fseek()** = posiciona o arquivo em um byte específico

Funções em C (stdio.h)

- **feof()** = retorna verdadeiro se atingiu o final do arquivo
- **rewind()** = coloca o ponteiro do arquivo no seu início
- **remove()** = apaga um arquivo
- **fflush()** = descarrega o conteúdo de um arquivo

Ponteiro para Arquivo

- Definição de variável do tipo arquivo:

  FILE *arq;

  arq é uma variável ponteiro capaz de identificar um arquivo no disco

  Ela **aponta para informações do arquivo:**

  - nome, status e posição do arquivo.

Criar e Abrir Arquivo

- Função **fopen(nome_arquivo, modo_abertura)**
  - Abre ou cria um arquivo, retornando o ponteiro apontado para o mesmo

- Exemplo:

  arq = fopen(nome_arquivo, modo_abertura);

Criar e Abrir Arquivo

- Função **fopen(nome_arquivo, modo_abertura)**

  nome_arquivo: string contendo o nome do arquivo para abrir ou criar, podendo incluir um path

  modo_abertura: string que representa como o arquivo será aberto: escrita, leitura ...

Criar e Abrir Arquivo

- **Modos de Abertura:**

w	w+	r	r+
a	a+	wb	wb+
rb	rb+	ab	ad+

Criar e Abrir Arquivo

- **Modos de Abertura:**

  w

  Cria um arquivo texto para escrita (apaga se ele já existir)

  w+

  Cria um arquivo texto para escrita e leitura (apaga se o arquivo já existir)

Criar e Abrir Arquivo

- **Modos de Abertura:**

  r

  Abre um arquivo texto para leitura

  r+

  Abre um arquivo texto para leitura e escrita (o arquivo deve existir)

Criar e Abrir Arquivo

- **Modos de Abertura:**

  a

  Abre um arquivo texto para anexar novos dados (no final)

  a+

  Anexa novos dados ou cria um arquivo texto para leitura e escrita (se o arquivo não existir, cria o arquivo)

Criar e Abrir Arquivo

- **Modos de Abertura:**

  wb

  Cria um arquivo binário para escrita (apaga se ele já existir)

  wb+

  Cria um arquivo binário para escrita e leitura (apaga se o arquivo já existir)

Criar e Abrir Arquivo

- **Modos de Abertura:**

  ab

  Abre um arquivo binário para anexar novos dados (no final)

  ab+

  Anexa novos dados ou cria um arquivo binário para leitura e escrita (se o arquivo não existir, cria o arquivo)

Criar e Abrir Arquivo

- **Exemplo:**

  FILE *arq1, *arq2;

  arq1 = fopen(“arquivo1.txt”, “w”);

  arq2 = fopen(“texto.txt”, “a+”);

Criar e Abrir Arquivo

- Importante verificar se o arquivo foi criado ou aberto com sucesso!

```
FILE *arq1, *arq2;
arq1 = fopen(“arquivo1.txt”, “w”);
if (arq1==NULL) {
    printf(“Erro na criação do arquivo);
    return(0);
}
else { ....}
```

Fechar Arquivo

- Função **fclose()**: fecha um arquivo
- É importante que todo arquivo aberto seja fechado antes de terminar o programa

  Exemplo: fclose(arq);

  arq: ponteiro para um arquivo obtido pela função fopen()

Arquivo – Modeo TEXTO

- **Gravação e Leitura:**

  função **fputs()**

  função **fgets()**

Arquivo – Modeo TEXTO

- função **fputs()**: escreve uma cadeia de caracteres em um arquivo

  Exemplo: fputs(cadeia_caracteres, arq)

- função **fgets()**: lê uma cadeia de caracteres até tam ou até encontrar \n

  Exemplo: fgets(cadeia_caracteres, tam, arq)

Funções de Manipulação de Arquivos

- Escrita de saída formatada: **fprintf()**
- Leitura de entrada formatada: **fscanf()**
- Insere uma string no arquivo: **fputs()**
- Lê uma string do arquivo: ***fgets()**

Arquivos TEXTO – Exemplo 01

- Abra o arquivo **CriarArquivo.c**
- Ele deve criar, em disco, o arquivo de nome **arquivo.txt**
- Após executá-lo, localize-o na pasta e dê um duplo-clique nele
- Verifique o seu conteúdo armazenado

Arquivos TEXTO – Exemplo 01

- **Salve** o arquivo como **CriarArquivoTeste.c**
- Altere a variável ponteiro para o arquivo
- Altere o nome do arquivo a ser criado em disco
- Execute-o
- Verifique o conteúdo do novo arquivo criado

Arquivos TEXTO – Exemplo 02

- Abra o arquivo **CriarArquivoVerifErro.c**
- Ele deve criar, em disco, o arquivo de nome **arquivo.txt**
- O detalhe é que agora ele **verifica** se ocorreu ou não algum **erro** na criação do arquivo
- Faça os testes e análises que julgar necessários

Arquivos TEXTO – Exemplo 03

- Abra o arquivo **GravarDadosArquivo.c**
- Ele deve criar, em disco, o arquivo de nome **arquivo_palavra.txt**
- Faça os testes e análises que julgar necessários

Arquivos TEXTO – Exemplo 03

- Salve o arquivo como **GravarDadosArquivo1.c**
- Ele deve criar, em disco, o arquivo de nome **arquivo_frase.txt**
- Faça os testes e análises que julgar necessários