Programação II (INF16153)

Aula 14: Arquivos binários

Prof. André Pacheco
http://pachecoandre.com.br/

Arquivos

Arquivos

- Arquivos são unidades que normalmente são armazenadas no disco (HD)
- Sistemas Operacionais permitem que arquivos sejam criados e recuperados por um nome em uma hierarquia de diretórios (pastas)
- Utilizamos arquivos em programação quando:
  - Não existe espaço na memória devido ao tamanho do dado
  - Há necessidade de persistir o dado
- Observação: em softwares profissionais a persistência é realizada por meio de **banco de dados**.
  - Mas também é possível realizar com arquivos (mas não usual e nem recomendado)
  - Porém, isso é tema para o seu eu do futuro

Arquivos

- A linguagem C permite o uso de dois tipos de arquivos: **texto** e **binário**
- **Arquivo de texto:**
  - Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de texto
  - Os dados são gravados como caracteres de 8 bits
    - Porque?

Arquivos

- A linguagem C permite o uso de dois tipos de arquivos: **texto** e **binário**
- **Arquivo de texto:**
  - Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de texto
  - Os dados são gravados como caracteres de 8 bits
    - Porque? **Tabela ASCII!**
- **Arquivo binário:**
  - Armazena uma sequência de bits que está sujeita às convenções dos programas que o gerou
    - Exemplo: arquivos executáveis
  - Os dados são gravados na forma binária (do mesmo modo que estão na memória)
    - Exemplo: um número inteiro de 32 bits com 8 dígitos ocupará 32 bits no arquivo

Manipulação de arquivos

Manipulando arquivos em C

- A linguagem C possui uma série de funções para manipular arquivos
  - A maioria das funções servem para binário ou não
  - Elas se encontram na biblioteca padrão de entrada e saída: `stdio.h`
- Basicamente, as operações básicas com arquivos são:
  - Abertura e fechamento
  - Leitura, escrita, alteração e/ou exclusão
- Em C, não existe uma função que ler automaticamente todos os dados do arquivo
  - Isso fica a cargo do programador
- Portanto, temos sempre que executar os 3 passos a seguir:
  - **Abrir ou criar o arquivo arquivo:** associar um nome físico a um lógico
  - **Manipular os dados:** ler, escrever, alterar e/ou excluir
  - **Fechar o arquivo**

Manipulando arquivos em C

- Todas as funções de manipulação de arquivo trabalham com o conceito de “**ponteiro para um arquivo**”
- Sempre que formos manipular um arquivo em C, precisamos criar um **ponteiro** para ele da seguinte forma:

```c
#include <stdio.h>

int main(){
    FILE *arq;
}
```

- O ponteiro `arq` nos permite manipular arquivos em C

Manipulando arquivos em C

- Todas as funções de manipulação de arquivo trabalham com o conceito de “**ponteiro para um arquivo**”
- Sempre que formos manipular um arquivo em C, precisamos criar um **ponteiro** para ele da seguinte forma:

```c
#include <stdio.h>

int main(){
    FILE *arq;
}
```

FILE é um tipo disponível dentro da biblioteca `stdio.h`

- O ponteiro `arq` nos permite manipular arquivos em C

Manipulando arquivos em C

- O primeiro passo para trabalhar com arquivos em C é a abertura
  - Isso é feito através da função `fopen` (*file open*)
  - **Assinatura:**
    - `FILE *fopen(char *nome_arq, char *modo)`

- Essa função recebe dois parâmetros:
  - **`nome_arq`:** é o nome do arquivo que vai ser mapeado no disco. Deve ser válido no sistema operacional alvo
    - Podemos usar um **caminho absoluto** ou **relativo**
      - **Caminho absoluto:** é o caminho completo até o arquivo. Exemplo: `/home/patcha/meu_arquivo.csv`
      - **Caminho relativo:** é apenas o nome do arquivo, o diretório usado é o mesmo de onde o programa foi chamado
  - **`modo`:** é o modo de abertura do arquivo. Veja a tabela no próximo slide

Manipulando arquivos em C

- **Modos de abertura de um arquivo:**

Modo	Arquivo	Função
"r"	Texto	Leitura. Arquivo deve existir.
"w"	Texto	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a"	Texto	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"rb"	Binário	Leitura. Arquivo deve existir.
"wb"	Binário	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"ab"	Binário	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+"	Texto	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+"	Texto	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+"	Texto	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+b"	Binário	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+b"	Binário	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+b"	Binário	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").

Manipulando arquivos em C

- **Modos de abertura de um arquivo:**

Para manipular arquivos binários, basta usar os modos que possuem a letra b

Modo	Arquivo	Função
"r"	Texto	Leitura. Arquivo deve existir.
"w"	Texto	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a"	Texto	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"rb"	Binário	Leitura. Arquivo deve existir.
"wb"	Binário	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"ab"	Binário	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+"	Texto	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+"	Texto	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+"	Texto	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+b"	Binário	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+b"	Binário	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+b"	Binário	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").

Manipulando arquivos em C

- Todavia, **sempre que abrimos um arquivo, temos que fechá-lo**
  - Por mais que seja o último passo, é uma boa prática sempre que abrir, também fechar
  - Isso evita que esqueçamos do fechamento
- Para fechar um arquivo, usamos a função `fclose` (*file close*)
  - **Assinatura:**
    - `int fclose(FILE *stream)`
  - Essa função retorna zero se tudo ocorreu corretamente

Arquivos binários

Arquivos binários

- Variáveis têm tamanho fixo na memória
  - Exemplo: um inteiro ocupa 4 bytes
- Representação em texto precisa de um número variável de dígitos, logo o tamanho também é variável
  - Exemplo: os números 50, 500, 5000, 50000 são armazenadas com tamanhos diferentes
- Armazenar dados em em arquivos de maneira análoga permite reduzir o tamanho do arquivo

Arquivos binários

- Arquivos binários em C são usados para armazenar e manipular dados no formato binários
  - Isso ocupa menos bytes do que um arquivo de texto
- Arquivos binários preservam os tipos originais
  - Útil para guardar dados mais complexos
- Uma desvantagem é que nós humanos somos incapazes de interpretar um arquivo binário visualmente

Leitura e escrita de arquivos binários

Leitura e escrita

- Para escrever dados em um arquivo binário, utilizamos a seguinte função da `stdio.h`:

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)
```

- Parâmetros:
  - `ptr`: ponteiro para o elemento a ser escrito
  - `size`: tamanho em bytes de cada elemento a ser escrito
  - `nmemb`: número de elementos que serão escritos
  - `stream`: ponteiro para o arquivo
- Retorno:
  - Quantidade de elementos escritos com sucesso

Leitura e escrita

- Para escrever dados em um arquivo binário, utilizamos a seguinte função da `stdio.h`:

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
```

- Parâmetros:
  - `ptr`: ponteiro para o elemento que vai salvar os dados
  - `size`: tamanho em bytes de cada elemento a ser lido
  - `nmemb`: número de elementos que serão lidos
  - `arq`: ponteiro para o arquivo
- Retorno:
  - Quantidade de elementos lidos com sucesso ou EOF

Leitura e escrita

- Ambas as funções retornam um valor do tipo `size_t`
  - Tipo de dado da biblioteca padrão usado para representar tamanho de objetos em bytes
- Se pegarmos esse retorno como um inteiro, ele é convertido a quantidade de bytes
- Isso é útil para identificarmos algum erro na escrita ou leitura
  - Se o retorno de `fwrite` for menor do que a quantidade de bytes solicitadas, isso significa um erro de escrita
  - Se o retorno de `fread` for menor do que a quantidade de bytes solicitadas, isso significa um erro de leitura ou que o final do arquivo foi atingido

Exemplos

Exemplo de escrita

```c
#include <stdio.h>

int main() {
    FILE *arquivo;
    arquivo = fopen("dados.bin", "wb");

    int numeros[] = {10, 20, 30, 40, 50};
    int elementos_gravados = fwrite(numeros, sizeof(int), 5, arquivo);

    printf("Elementos gravados: %d\n", elementos_gravados);

    fclose(arquivo);
    return 0;
}
```

Exemplo de escrita

```c
#include <stdio.h>
int main() {
    FILE *arquivo;
    arquivo = fopen("dados.bin", "rb");
    int numeros[5];
    int elementos_lidos = fread(numeros, sizeof(int), 5, arquivo);
    printf("Elementos lidos: %d\n", elementos_lidos);
    for (int i = 0; i < elementos_lidos; i++) {
        printf("%d ", numeros[i]);
    }
    printf("\n");
    fclose(arquivo);
    return 0;
}
```

Exemplo de escrita

```c
#include <stdio.h>
int main() {
    FILE *arquivo;
    arquivo = fopen("dados.bin", "rb");
    int numeros[5];
    int elementos_lidos = fread(numeros, sizeof(int), 5, arquivo);
    printf("Elementos lidos: %d\n", elementos_lidos);
    for (int i = 0; i < elementos_lidos; i++) {
        printf("%d ", numeros[i]);
    }
    printf("\n");
    fclose(arquivo);
    return 0;
}
```

É necessário conhecer como os bytes foram armazenados para saber reconstruir o arquivo

Vantagens e desvantagens

Vantagens de arquivos binários

- São geralmente mais eficientes em termos de espaço de armazenamento e velocidade de acesso em comparação com arquivos de texto
  - Não há necessidade de conversão ou processamento adicional ao ler ou escrever os dados
- Permitem armazenar qualquer tipo de dado, incluindo dados personalizados e complexos, como matrizes multidimensionais e estruturas

Desvantagens de arquivos binários

- Podem não ser facilmente portáveis entre diferentes plataformas ou sistemas operacionais
  - Tamanho de armazenamento pode divergir
  - Tem que ter mais cuidado ao compartilhar
- Não são legíveis para humanos
  - Mais difícil interpretar os dados e depurar
- Qualquer alteração na estrutura dos dados ou no formato de armazenamento pode corromper o arquivo inteiro

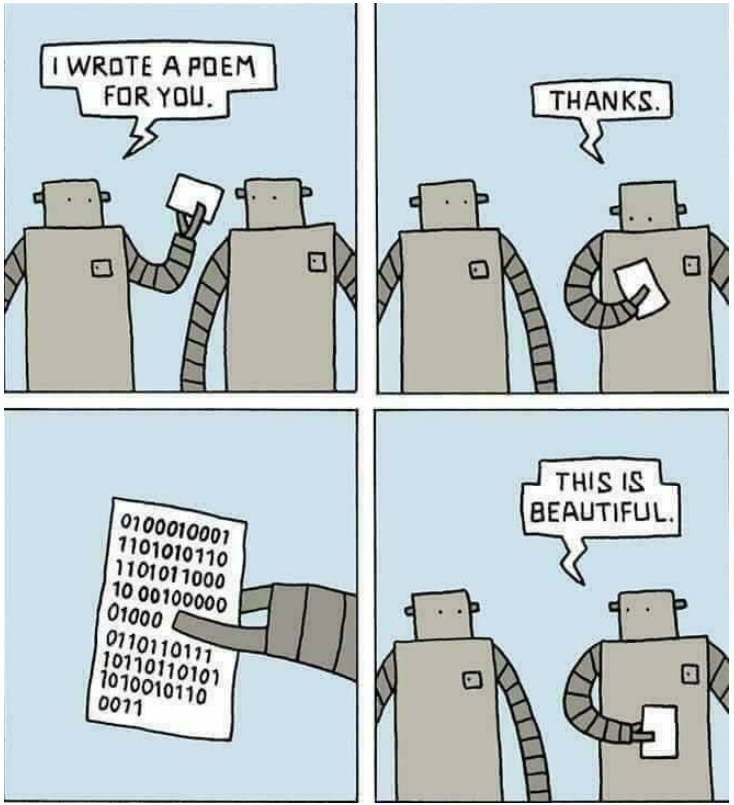
Exercícios

Exercício 1

- Crie uma estrutura chamada pessoa, que armazena nome, data de nascimento e cpf
- Crie e inicialize um array de pessoas
- Crie uma função para salvar este array de pessoas em um arquivo binário
- Crie uma função para ler o array de pessoas a partir do arquivo binário

Exercício 2

- O arquivo disponível neste link representa uma imagem
- Faça uma função que leia este arquivo e imprima na tela maior, o menor e o pixel médio
- Os dados da imagem estão organizados da seguinte forma:
  - Dois inteiros de 4 bytes contendo o número de linhas M e de colunas N da imagem
  - Um inteiro de 4 bytes representando o tipo da imagem (pode desconsiderar)
  - M * N valores dos tipos int com 4 bytes cada representando os pixels



Obrigado!

Contato:
@paaatcha
apacheco@inf.ufes.br
github.com/paaatcha
linkedin.com/in/pacheco-andre/
pachecoandre.com.br