

Arquivos

Prof. Marcelo Rosa

Algoritmos e Estrutura de Dados 2 (AE43CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

1 Introdução

2 Tipos de Arquivos

- Arquivos texto
- Arquivos binários

3 Stream

4 Arquivos

- Abertura e fechamento
- Modo de processamento

Introdução

- O armazenamento de dados na memória RAM é temporário
- Arquivos são utilizados para o armazenamento de dados em dispositivos secundários (HD, pendrive, cartão de memória, CD, DVD, etc)
- Sistema de arquivos¹



Introdução

Arquivo

- Coleção de bytes armazenados em um dispositivo de armazenamento secundário
- Dados, programas, etc
- Arquivos de texto e binários
- Gerenciados pelo sistema operacional

Vantagens de se usar arquivos:

- Armazenamento **durável** (não se perde ao desligar o dispositivo)
- Permitem **armazenar uma grande quantidade de informação**
- **Acesso concorrente** aos dados (ou seja, pode ser usado por vários processos)

Cuidado

- A **extensão** do arquivo **não define o seu tipo**
 - A extensão só serve para indicar para o SO qual é o programa mais indicado para abrir aquele arquivo
- O que define um arquivo é a **maneira como os dados estão organizados e as operações usadas por um programa para processar** (ler ou escrever) esse arquivo.

- Atributos de arquivos
 - Nome
 - Identificador
 - Tipo
 - Posição
 - Tamanho
 - Proteção
 - Hora, data e identificação do usuário

Biblioteca em C para manipular arquivos

- Uma biblioteca da linguagem C define um conjunto completo de funções de entrada/saída
 - `#include <stdio.h>`
- A linguagem C usa um **tipo especial de ponteiro** para manipular arquivos

```
1 //Forma geral
2 FILE *nome_ponteiro;
```

- É esse ponteiro que controla o fluxo de leitura e escrita dentro de um arquivo
 - Para leitura ou escrita do arquivo será necessário que esse ponteiro seja definido antes.

1 Introdução

2 Tipos de Arquivos

- Arquivos texto
- Arquivos binários

3 Stream

4 Arquivos

- Abertura e fechamento
- Modo de processamento

Tipos de Arquivos: texto e binários

A linguagem C trabalha com apenas dois tipos de arquivos:

- **Arquivos texto:** podem ser editados no bloco de notas
- **Arquivos binários:** **Não** podem ser editados no bloco de notas

1 Introdução

2 Tipos de Arquivos

- Arquivos texto
- Arquivos binários

3 Stream

4 Arquivos

- Abertura e fechamento
- Modo de processamento

Tipos de Arquivos: texto e binários

Arquivo de texto

- Os **dados** são gravados exatamente como **seriam impressos na tela**
- Os **dados** são gravados como **caracteres de 8 bits** / 1 byte utilizando a **tabela ASCII**
- Para isso, existem uma etapa de “conversão dos dados”

Problemas com a conversão

- arquivos maiores
- leitura e escrita lentas

Exemplo

- Considere um número inteiro com 8 dígitos

```
1 int n = 12345678; // 32 bits ou 4 bytes na memória
```

- Num arquivo texto, cada dígito será convertido para seu caractere ASCII, ou seja, 8 bits por dígitos

```
1 "12345678" // 64 bits ou 8 bytes no arquivo
```

1 Introdução

2 Tipos de Arquivos

- Arquivos texto
- Arquivos binários

3 Stream

4 Arquivos

- Abertura e fechamento
- Modo de processamento

Tipos de Arquivos: texto e binários

Arquivo de binário

- Os **dados** são gravados exatamente como **estão organizados na memória** do computador
- Não existe etapa de “conversão” dos dados.
- Consequentemente
 - arquivos em geral menores
 - leitura e escrita mais rápidas

Exemplo

- Voltemos ao número inteiro com 8 dígitos

```
1      int n = 12345678; // 32 bits ou 4 bytes na memória
```

- Num arquivo binário, o conteúdo da memória, em bits, será copiado diretamente para o arquivo, sem conversão

```
1      12345678 // 32 bits ou 4 bytes no arquivo (codificado)
```

1 Introdução

2 Tipos de Arquivos

- Arquivos texto
- Arquivos binários

3 Stream

4 Arquivos

- Abertura e fechamento
- Modo de processamento

- O sistema de arquivos C é projetado para possibilitar o uso de vários dispositivos (discos, impressoras, teclados)
- O sistema de arquivos transforma os dados carregados em um dispositivo lógico denominado *stream*
- Apesar dos dispositivos de entrada/saída serem distintos, todas as *streams* comportam-se de forma similar, ou seja, são amplamente independentes de dispositivo
- Existem dois tipos de *streams*
 - *Stream* do tipo texto: sequência de caracteres
 - *Stream* binária: sequência de bytes

1 Introdução

2 Tipos de Arquivos

- Arquivos texto
- Arquivos binários

3 Stream

4 Arquivos

- Abertura e fechamento
- Modo de processamento

1 Introdução

2 Tipos de Arquivos

- Arquivos texto
- Arquivos binários

3 Stream

4 Arquivos

- Abertura e fechamento
- Modo de processamento

Arquivos

Abrir um arquivo

- Na linguagem C, um arquivo é uma sequência de *bytes*
- Quanto um arquivo é aberto, um ponteiro é associado à sua respectiva *stream*
- Os arquivos são manipulados por meio de ponteiros (`FILE *`)

Arquivos

Abrir um arquivo

Função fopen()

- Permite abrir um arquivo em um determinado modo de leitura ou escrita

Forma geral

1

```
File* fopen(char *nome_arquivo, char *modo)
```

- nome_arquivo: nome (e caminho, se necessário) do arquivo a ser aberto.
- modo: define como o arquivo será aberto (leitura, escrita, binário, etc).

Arquivos

Abrir um arquivo

Função fopen()

- Permite abrir um arquivo em um determinado modo de leitura ou escrita

Forma geral

```
1 File* fopen(char *nome_arquivo, char *modo)
```

- nome_arquivo: nome (e caminho, se necessário) do arquivo a ser aberto.
- modo: define como o arquivo será aberto (leitura, escrita, binário, etc).

Exemplo

```
1 File *f = fopen("arquivo.txt", "w");
```

Arquivos

Abrir um arquivo

Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      /*Se a função fopen() não conseguir
6      abrir o arquivo, ela irá retornar NULL
7      */
8
9      // Abre o arquivo texto para escrita ("w")
10     FILE *f = fopen("G:\\dados.txt", "w");
11
12     if (f == NULL) {
13         printf("Erro ao abrir o arquivo!\n");
14         return 1;
15     }
16     return 0;
17 }
```

Caminho

Para o “nome” do arquivo, podemos usar o caminho

- **absoluto**: endereço completo
- **relativo**: relativo a pasta do programa

Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      FILE *f;
6
7      //Caminho absoluto
8      f = fopen("G:\\projetos\\dados.txt", "w");
9
10     //Caminho relativo
11     f = fopen("arquivo.txt", "w");
12     f = fopen("../novo\\arquivo2.txt", "w");
13
14     return 0;
15 }
```

1 Introdução

2 Tipos de Arquivos

- Arquivos texto
- Arquivos binários

3 Stream

4 Arquivos

- Abertura e fechamento
- Modo de processamento

Modos de processamento de arquivos do tipo texto

- "r": abre um arquivo texto para leitura
- "w": abre um arquivo texto para escrita. Caso o arquivo não exista, o mesmo será criado, caso contrário, o conteúdo do mesmo é apagado
- "a": abre um arquivo texto para escrita no final
- "r+": abre um arquivo texto para leitura e gravação (o arquivo deve existir e pode ser alterado)
- "w+": abre um arquivo texto para leitura e gravação. Se o arquivo não existir, o mesmo é criado, caso contrário o conteúdo do mesmo é apagado
- "a+": abre um arquivo texto para leitura e gravação. Os dados são adicionados no final do arquivo

Arquivos

Modo de processamento: arquivo binário

Modos de processamento de arquivos do tipo binário

- "rb": abre um arquivo binário para leitura
- "wb": abre um arquivo binário para escrita. Caso o arquivo não exista, o mesmo será criado, caso contrário, o conteúdo do mesmo é apagado
- "ab": abre um arquivo binário para escrita no final
- "r+b" ou "rb+": abre um arquivo binário para leitura e gravação (o arquivo deve existir e pode ser alterado)
- "w+b" ou "wb+": abre um arquivo texto para leitura e gravação. Se o arquivo não existir, o mesmo é criado, caso contrário o conteúdo do mesmo é apagado
- "a+b" ou "ab+": abre um arquivo texto para leitura e gravação. Os dados são adicionados no final do arquivo

Arquivos

Fechar um arquivo

função `fclose()`

- Após o uso do arquivo, o mesmo deve ser fechado por meio da função `fclose()`

Formal geral

```
1 int fclose(File *f);
```

- A função `fclose` retorna 0, caso a operação seja bem-sucedida, ou 1, caso ocorra um erro na operação

Arquivos

Funções principais para o processamento de arquivos texto

fgetc() e getc()

- Permite a leitura de um caractere por vez de um arquivo texto

Formal geral

```
1      int fgetc(File *f);  
2      int getc(File *f);
```

- A função fgetc() lê o próximo caractere do arquivo e retorna ele como um inteiro, ou EOF símbolo de final de arquivo (-1)
- A função getc() é equivalente a fgetc(), mas pode ser implementa como macro.

Arquivos

Funções principais para o processamento de arquivos texto

fputc() e putc()

- escreve caracteres no arquivo texto

Formal geral

```
1      int fputc(int c, File *f);  
2      int putc(int c, File *f);
```

- A função fputc() escreve o caractere c, convertido em char, no arquivo
- A função putc() é equivalente a fgetc(), mas pode ser implementa como macro.
- Ambas retornam o caractere escrito convertido em int ou EOF em caso de erro.

Arquivos

Funções principais para o processamento de arquivos texto

fgets()

- Permite ler uma string de tamanho n no arquivo

Formal geral

```
1 char * fgets(char * s, int n, FILE * arquivo)
```

- A função `fgets` lê uma linha de texto de um arquivo , armazenando os caracteres em um buffer (ponteiro `s`).
- Ela retorna o ponteiro `s` em caso de sucesso, e `NULL` em caso de erro ou fim de arquivo.

Arquivos

Funções principais para o processamento de arquivos texto

fputs()

- Permite escrever uma string no arquivo

Formal geral

```
1 char * fputs(char * s, FILE * arquivo)
```

- A função fputs escreve a string s no arquivo, sem o \0
- Ela retorna um número inteiro não negado em caso de sucesso, e EOF em caso de erro.

Arquivos

Funções principais para o processamento de arquivos texto

feof()

- Permite verificar se o fim de um arquivo (EOF) foi alcançado.

Formal geral

```
1 int feof( FILE * arquivo)
```

- A função feof retorna um número diferente de zero se o fim foi atingido, ou zero se ainda há dados para serem lidos.

Arquivos

Funções principais para o processamento de arquivos binário

fread()

- Permite ler o conteúdo de um arquivo binário

Formal geral

```
1 int fread(void * buffer, int qtd_bytes, int n_unidades, FILE * arquivo)
```

- buffer: região da memória onde os dados são armazenados
- qtd_bytes: número de bytes que deverão ser lidos por unidade
- n_unidade: quantidade de unidades que deverão ser lidas
- Essa função retorna a quantidade de unidades lidas efetivamente

Arquivos

Funções principais para o processamento de arquivos binário

`fwrite()`

- Permite escrever dados em um arquivo binário

Formal geral

```
1 int fwrite(void * buffer, int qtd_bytes, int n_unidades, FILE * arquivo)
```

- Essa função é simétrica a `fread`

Arquivos

Funções principais para o processamento de arquivos binário

`fseek()`

- Permite mover o ponteiro interno do arquivo para uma posição específica, permitindo a leitura ou escrita em qualquer parte do arquivo.

Arquivos

Funções principais para o processamento de arquivos binário

Formal geral

```
1 int fseek(FILE * arquivo, int qtd_bytes, int posicao_origem)
```

- move a posição do cursor (*qtd_bytes*) a partir de um localização específica (*posicao_origem*)
- Na variável *posicao_origem* pode ser atribuído os seguintes valores
 - SEEK_SET: a partir da posição inicial do arquivo (corresponde ao valor 0)
 - SEEK_CUR: a partir da posição atual (corresponde ao valor 1)
 - SEEK_END: a partir do final do arquivo (corresponde ao valor 2)

Arquivos

Funções principais para o processamento de arquivos binário

Exemplo

- Por exemplo, para mover para o início do arquivo:

```
1 fseek(f, 0, SEEK_SET);
```

- Para pular 10 bytes à frente da posição atual:

```
1 fseek(f, 10, SEEK_CUR);
```

- Para ir 5 bytes antes do fim do arquivo:

```
1 fseek(f, -5, SEEK_END);
```

Arquivos


Funções principais para o processamento de arquivos binário

rewind()

- Permite retornar o cursor ao início do arquivo

Formal geral

```
1 void rewind(FILE * arquivo)
```

 Deitel, H. M. e and Deitel, P. J.
C: Como Programar.
Pearson, 2011.

 Schidildt, H.
C Completo e Total.
Pearson, 2011.