Engenharia de Computação

Fundamentos de Programação

Aula 21 – Arquivos – Parte 1

Prof. Muriel de Souza Godoi
muriel@utfpr.edu.br

Arquivos

- O que são arquivos?
  - Arquivos são **coleções de dados armazenados em um dispositivo de armazenamento secundário**
    - Fora da memória principal (RAM);
  - Exemplos de memória secundária:
    - disco rígido
    - pen-drive;

Arquivos

- Por que usar arquivos no **HD**?
  - Permitem armazenar **grande quantidade** de informação;
  - **Persistência** dos dados (disco);
  - Acesso aos dados poder ser **não sequencial**;
  - **Acesso concorrente** aos dados
    - Mais de um programa pode usar os dados ao mesmo tempo

Tipos de Arquivos

- A **Linguagem C** trabalha com dois tipos de arquivos: de texto e binários.
- **Arquivo texto**
  - Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de textos simples como um Bloco de Notas.
  - Os dados são gravados como caracteres de **8 bits**.
    - **Ex.:** Um número inteiro de 32 bits com 8 dígitos ocupará **64 bits** no arquivo (8 bits por dígito).

```
char nome[20]="Rosinosberta";
int i = 32;
float a = 1.75;
```

Editor de Texto

```
Rosinosberta 32 1.750000
```

Tipos de Arquivos

- **Arquivo binário**
  - Armazena uma sequência de bits que está sujeita as convenções dos programas que o gerou.
    - **Ex:** arquivos executáveis, arquivos compactados, arquivos de registros, etc.
  - Os dados são gravados na forma binária (do mesmo modo que estão na memória).
    - **Ex.:** um número inteiro de 32 bits com 8 dígitos ocupará 32 bits no arquivo.

```
char nome[20]=“Rosinosberta”;
int i = 32;
float a = 1.75;
```

Editor de Texto

```
Rosinosberta £? ¬¢
```

Manipulando arquivos em C

- A **Linguagem C** possui uma série de funções para manipulação de arquivos, cujos protótipos estão reunidos na biblioteca padrão de estrada e saída, **stdio** (**St**an**d**ard **I**nput **O**utput):

```
#include <stdio.h>
```

  - A linguagem **não possui funções automáticas** que leem todas as informações de um arquivo;
  - Suas funções se limitam a abrir e fechar, ler e escrever caracteres / bytes;
  - É **tarefa do programador** criar a função que lerá um arquivo de uma maneira específica.

Manipulando arquivos em C

- Todas as funções de manipulação de arquivos trabalham com o conceito de **ponteiro de arquivo**
  - Podemos declarar um ponteiro de arquivo da seguinte maneira:

```
FILE *fp;
```

- **fp** (*file pointer*) é o ponteiro para arquivos que nos permitirá manipular arquivos no C.

Abrindo um arquivo

- Para a abertura de um arquivo, usa-se a função **fopen**

```
FILE* fopen(char* nome_arquivo, char* modo);
```

  - O parâmetro **nome_arquivo** determina qual arquivo deverá ser aberto
    - Ele deve ser válido no sistema operacional que estiver sendo utilizado.
  - O **modo** de abertura determina que tipo de uso será feito do arquivo (leitura, escrita, outros…)
  - Retorna **NULL** se ocorrer erro na abertura

Abrindo um arquivo

- No parâmetro nome_arquivo pode-se trabalhar com caminhos absolutos ou relativos.
  - **Caminho absoluto:** descrição de um caminho desde o diretório raiz.
    - **Windows** - `"C:\\Projetos\\dados.txt"`
    - **Linux** - `"/home/usuario/dados.txt"`
  - **Caminho relativo:** descrição de um caminho desde o diretório corrente (onde o executável está salvo)
    - **Pasta atual** – `"arq.txt"`
    - **Subpasta** - `"diretorio/dados.txt"`
    - **Pasta acima** - `"../dados.txt"`

Abrindo um arquivo

- 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").

Abrindo um arquivo

- Outros modos de abertura de um arquivo

Modo	Arquivo	Função
"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").

Abrindo um arquivo

- Um arquivo texto pode ser aberto para escrita utilizando o seguinte conjunto de comandos:

```
FILE *arq = fopen("dados.txt","w");

if(arq == NULL){
  printf("Erro ao abrir o arquivo!\n");
  exit(1);// 1 significa erro
}// if
fclose(arq);
```

  - Caso o arquivo não tenha sido aberto com sucesso, programa não poderá continuar a executar;
  - Nesse caso, utilizamos a função **`exit()`**, presente na biblioteca `stdlib.h`, para abortar o programa a partir de qualquer ponto

Fechando um arquivo

- Sempre que terminamos de usar um arquivo que abrimos, devemos fechá-lo;
- Para isso usa-se a função **fclose**

```
int fclose(FILE* arq);
```

  - O ponteiro **arq** passado à função **fclose** determina o arquivo a ser fechado.
  - A função retorna zero no caso de sucesso.
  - Ao fechar um arquivo, todo caractere que tenha permanecido no **buffer** é gravado.

Fechando um arquivo

- O que é um buffer?
  - O **buffer** é uma região de memória que armazena **temporariamente** os dados enquanto eles estão sendo movidos de um lugar para outro (**Ex.:** RAM para HD)
- Por que utilizar?
  - Para ler e escrever arquivos no disco temos que **posicionar a cabeça** de gravação em um ponto específico do disco.
  - Se tivéssemos que fazer isso para cada caractere lido/escrito, a leitura/escrita de um arquivo seria uma operação **muita lenta**.
  - Assim a gravação só é realizada quando há um **volume razoável de informações** a serem gravadas ou quando o arquivo for fechado.

Escrita/Leitura em Arquivos

- Uma vez aberto um arquivo, podemos ler ou escrever nele.
- Para tanto, a Linguagem C conta com uma série de funções de leitura/escrita que variam de funcionalidade para atender as diversas aplicações:
  - Ler / Escrever um caractere
  - Ler / Escrever uma string
  - Ler / Escrever uma sequencia de bytes
  - Ler / Escrever uma string formatada

Escrita/Leitura de Caracteres

- A maneira mais fácil de se trabalhar com um arquivo é a leitura/escrita de um **único caractere**
- A função **fputc** escreve o caractere **ch** no arquivo

```
int fputc(int ch, FILE* arq);
```

- A função **fgetc** lê um caractere do arquivo

```
int fgetc(FILE* arq);
```

  - Após a leitura, o ponteiro do arquivo avança para o próximo caractere
  - Se não houver valor a ser lido, retorna **-1** (EOF).
    - Por isso, a função retorna **int** e não **char**

Escrita/Leitura de Caracteres

- Exemplo da função **fputc**

```c
FILE *arq;
char texto[100] = "Texto que sera escrito no arquivo";
// Abre o arquivo
arq = fopen("arquivo.txt","r");
if(arq == NULL){
  printf("Erro na abertura do arquivo");
  exit(1); //finaliza o programa
}//if
// Percorre o vetor escrevendo no arquivo
for (int i = 0; i < strlen(texto); i++) {
  fputc(texto[i],arq);
}//for
//Fecha o arquivo
fclose(arq);
```

Escrita/Leitura de Caracteres

- Exemplo da função **fgetc**

```
FILE *arq;
char c;
// Abre o arquivo
arq = fopen("arquivo.txt","r");
if(arq == NULL){
  printf("Erro na abertura do arquivo");
  exit(1); //finaliza o programa
}//if
// Lê enquanto não chegar no fim do arquivo
while ((c = fgetc(arq)) != EOF) {
  printf("%c",c);
}//for
//Fecha o arquivo
fclose(arq);
```

Arquivos pré-definidos

- A Linguagem C tem alguns arquivos **pré-definidos**
  - **`stdin`**
    - dispositivo de entrada padrão (geralmente o teclado)
  - **`stdout`**
    - dispositivo de saída padrão (geralmente o vídeo)
  - **`stderr`**
    - dispositivo de saída de erro padrão (geralmente o vídeo)
  - **`stdaux`**
    - dispositivo de saída auxiliar (em muitos sistemas, associado à porta serial)
  - **`stdprn`**
    - dispositivo de impressão padrão (em muitos sistemas, associado à porta paralela)

Arquivos pré-definidos

- Assim, podemos utilizar a função **fputc** para escrever um caractere no terminal
  - Dispositivo de **saída** padrão

```
fputc(caractere, stdout);
```

- E também podemos utilizar a função **fgetc** para ler um caractere do teclado
  - Dispositivo de **entrada** padrão

```
caractere = fgetc(stdin);
```

Escrita/Leitura de Strings

- Também é possível de trabalhar com um arquivo por meio da leitura/escrita de **sequencia de caracteres**
- A função **fputs** escreve a string **str** no arquivo

```
int fputs(char *str, FILE* arq);
```

- A função **fgets** lê uma string do arquivo

```
char* fgets(char *str, int tamanho, FILE* arq);
```

  - **str**: onde a string lida será armazenada;
  - **tamanho**: número máximo de caracteres a serem lidos;
  - Se não houver valor a ser lido, retorna **NULL**.
    - Ponteiro para o início do vetor caso sucesso

Escrita/Leitura de Strings

- Exemplo da função **fputs**:

```c
FILE *arq;
char texto[100] = "Texto que sera escrito via fputs";
int resultado;

// Abre o arquivo
arq = fopen("arquivo.txt","w");
if(arq == NULL){
  printf("Erro na abertura do arquivo\n");
  exit(1); //finaliza o programa
}//if

fputs(texto, arq);// Escreve o texto do arquivo
if (resultado == EOF) {
  printf("Erro na gravação\n");
}//for
fclose(arq); //Fecha o arquivo
```

Escrita/Leitura de Strings

- Exemplo da função **fgets**:

```
FILE *arq;
char texto[100];
char* resultado;

arq = fopen("arquivo.txt","r"); // Abre o arquivo
if(arq == NULL){
  printf("Erro na abertura do arquivo\n");
  exit(1); //finaliza o programa
}//if

fgets(texto, 100,arq);// Lê o texto do arquivo
if (resultado == EOF) {
  printf("Erro na leitura\n");
}else{
 printf("%s\n",texto);
}//else
fclose(arq);//Fecha o arquivo
```

Escrita/Leitura de Strings

- Assim como no caso anterior, podemos utilizar a função **fputc** para escrever uma string caractere no terminal
  - Dispositivo de **saída** padrão

```
char string[30] = "Hello World";
fputs(string, stdout);
```

- E também podemos utilizar a função **fgets** para ler uma string do teclado (já vimos isso!)
  - Dispositivo de **entrada** padrão

```
char nome[30];
fgets(nome, 30, stdin);
```

Exercícios

- **1)** Escreva um programa que leia do usuário o nome de um arquivo texto. Em seguida, mostre na tela quantas linhas esse arquivo possui.
- **2)** Escreva um programa que leia do usuário os nomes de dois arquivos texto. Crie um terceiro arquivo texto com o conteúdo dos dois primeiros juntos (o conteúdo do primeiro seguido do conteúdo do segundo).
- **3)** Escreva um programa para converter o conteúdo de um arquivo texto em caracteres maiúsculos. O programa deverá ler do usuário o nome do arquivo a ser convertido e o nome do arquivo a ser salvo.
- **4)** Crie um programa para calcular e exibir o número de palavras contido em um arquivo texto. O usuário deverá informar o nome do arquivo.

Exercícios

- **5)** Elabore um programa para calcular e exibir o número de vezes que cada letra ocorre dentro de um arquivo texto. Ignore as letras com acento. O usuário deverá informar o nome do arquivo.
- **6)** Elabore um programa no qual o usuário informe o nome de um arquivo texto e uma palavra, e o programa informe o número de vezes que aquela palavra aparece