Aula 11: Arquivos

Professor(a): João Eduardo Montandon (103)
Virgínia Fernandes Mota (106)
jemaf.github.io
http://www.dcc.ufmg.br/∼virginiaferm

INTRODUÇÃO A PROGRAMAÇÃO - SETOR DE INFORMÁTICA

Introdução

- Arquivo é um recurso que o sistema operacional promove e que as linguagens de programação tem acesso.
- É uma abstração que simplifica a criação e manipulação de dispositivos de entrada e saída de dados, não apenas restringindo a arquivos físicos gravados no disco rígido, mas também dispositivos como o teclado e o monitor.

Introdução

- Quando criamos um arquivo, seja através de uma linguagem de programação ou através da interface de usuário, na verdade não estamos propriamente criando o arquivo, mas sim delegando esta tarefa ao Sistema Operacional (Linux, Windows, Mac).
- Este se encarrega de fazer toda a verificação de permissões do usuário que está executando esta tarefa e atribuindo às descrições do arquivo informações como: usuário dono, data de criação, tipo, tamanho, data de acesso, modificação, dentre outras.

Tipos de arquivos

- Os arquivos são divididos em duas subcategorias:
  1. Binários: Arquivos interpretados apenas pelo SO. Contém apenas 0s e 1s. Programas compilados e bibliotecas são exemplos de arquivos binários;
  2. Textos: São arquivos que contém informações, não apenas se restringindo a texto propriamente dito, mas também a arquivos de mídia por exemplo.

Operações básicas

- O processo de utilização de um arquivo envolve, no mínimo, três etapas:
  - criação ou abertura do arquivo;
  - leitura ou escrita de dados no arquivo;
  - fechamento do arquivo.

Criação de arquivos

- Na primeira etapa, se o arquivo ainda não existir, ele deve ser criado.
- Caso o arquivo já exista (pelo fato de ter sido criado em uma execução anterior do programa) ele pode ser aberto para que novos dados sejam acrescentados ou para que os dados guardados nele possam ser lidos.

Leitura ou escrita de dados

- A segunda etapa é a que efetivamente lê ou grava dados no arquivo.
- A linguagem C oferece funções específicas de leitura e de escrita de dados em um arquivo.
  - Se o arquivo for do tipo texto (como este que você está lendo), você deve utilizar funções específicas para arquivos-texto.
  - Se ele for binário (do tipo que armazena registros ou estruturas), você deve utilizar as funções de leitura e/ou escrita em arquivos binários.

Fechamento

- A terceira etapa consiste em fechar o arquivo, para que seus dados sejam efetivamente gravados e fiquem protegidos até que o arquivo seja aberto novamente.

Manipulando arquivos em C

- Em C, para poder trabalhar em um arquivo, precisamos abrí-lo, associando-o a uma variável interna do programa. Para isso, usamos variáveis do tipo FILE *, cuja declaração é:
  FILE *arquivo
- Todas as funções de manipulação de arquivo necessitam de uma variável deste tipo para poder manipular esse arquivo.
- Depois de ter declarado uma variável que vai representar o arquivo, você deve efetivamente tentar **abrir** ou **criar** o arquivo!

Manipulando arquivos em C - Função **fopen()**

- A associação entre variável e arquivo é feita pela função **fopen()**. É ela quem abrirá o arquivo:
  variavel = fopen("nome_arquivo.extensao", "modo");
  - O primeiro parâmetro é o local onde o arquivo se encontra ou se for criado, onde o arquivo deve ser armazenado.
  - O segundo parâmetro especifica como o arquivo deve ser aberto.
  - Para abrir um arquivo o modo deve ser "r", e para criar um arquivo (ou sobrescrever um já existente) o modo deve ser "w".
  - Cuidado com o modo "w", pois todos os dados de um arquivo são apagados, caso ele já exista.
  - Outros modos estão especificados na tabela a seguir.

Manipulando arquivos em C - Função **fopen()**

r	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
w	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
a	Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
rb	Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário.
wb	Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário.
ab	Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário.
r+	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
w+	Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
a+	Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
r+b	Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, só que o arquivo é binário.
w+b	Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, só que o arquivo é binário.
a+b	Acrescenta dados ou cria uma arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário.

Manipulando arquivos em C - Função **fopen()**

- Você deve testar se o arquivo foi realmente aberto ou criado.
- Para saber se a função **fopen** realmente criou ou abriu o arquivo que você solicitou, teste se a variável que representa o arquivo possui algum valor dentro dela.

```
1 if (arquivo != NULL){
2     coloque as funções de manipulação de arquivo aqui dentro
3
4 }
```

Exemplo

```
1  #include <stdio.h>
2  int main(){
3      FILE *fp;
4      fp = fopen ("arquivo.txt","w");
5      if (!fp) // fp == NULL
6          printf ("Erro na abertura do arquivo.");
7      else
8          printf("Arquivo aberto com sucesso.");
9
10     fclose(fp);
11     return 0;
12 }
```

Manipulando arquivos em C - Funções **fprintf** e **fscanf**

- Depois de ter aberto ou criado um arquivo, e ter uma variável que o represente dentro do programa, você pode começar a gravar ou a ler dados neste arquivo.
- **Gravar ou escrever**: função **fprintf**, que é muito parecida com a função printf. A única diferença é que há um parâmetro (o primeiro) indicando qual é o arquivo onde os dados serão gravados (ou melhor, impressos).
- **Ler**: função **fscanf**, que é muito parecida com a função scanf. A única diferença é que há um parâmetro (o primeiro) indicando qual é o arquivo onde os dados serão lidos.

Manipulando arquivos em C - Funções **fprintf** e **fscanf**

- Exemplos:

```
1       FILE *entrada;
2       FILE *saida;
3       fprintf(saida, "soma = %10.6f quociente = %10.6f\n", soma,
            quociente);
4       fscanf(entrada, "%d %d", &n1, &n2);}
```

- Importante:
  - Quando você lê ou grava alguma coisa do arquivo, automaticamente você é colocado na próxima posição do mesmo
  - Esta é a posição de onde será lido ou escrito o próximo dado.
  - Normalmente, em um acesso sequencial a um arquivo, não temos que mexer nesta posição pois quando lemos um dado a posição no arquivo é automaticamente atualizada.

Manipulando arquivos em C - Função **feof**

- Não se pode ler nada após o fim do arquivo!
- O marcador de fim de arquivo é denominado "EOF"(end of file).
- O comando **feof(arquivo)** retorna verdadeiro quando a posição do arquivo atinge o "EOF". Dessa forma, podemos ler todo o conteúdo de um arquivo usando o comando fscanf dentro de um loop while(!feof(arquivo)).

Manipulando arquivos em C - Função **fclose**

- Quando acabamos de usar um arquivo que abrimos, devemos fechá-lo. Para fechar um arquivo basta chamar a função fclose: **fclose(arquivo);**
- O ponteiro "arquivo"determina o arquivo a ser fechado. A função retorna zero no caso de sucesso.
- Fechar um arquivo faz com que qualquer dado que tenha permanecido no "buffer"associado ao fluxo de saída seja gravado.
- A função exit() fecha todos os arquivos que um programa tiver aberto.

Exemplo

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

int main(){

    FILE *notas;
    char c, str[40] = "teste.txt", frase[80];
    int i = 0, n, n1;
    if (!(notas = fopen(str,"w"))){
        printf("Erro! Impossivel abrir o arquivo!\n");
        exit(1);
    }

    printf("Digite o numero de alunos");
    scanf("%d", &n);

    //(...)
```

Exemplo

```
1     //(...)
2     do{
3          fprintf(notas, "\n");
4          printf("\nDigite o primeiro nome do aluno: ");
5          scanf("%s", frase);
6          printf("\nDigite a nota: ");
7          scanf("%d", &n1);
8          fprintf(notas, "%s %d", frase, n1);
9          i++;
10    }while (i < n);
11
12    fclose(notas);
13    //(...)
```

Exemplo

```
1     //(...)
2     notas = fopen(str,"r");
3     printf("Eu li isso no arquivo:\n");
4     while (!feof(notas)){
5         fscanf(notas, "%s %d", frase, &n1);
6         printf("%s %d\n", frase, n1);
7     }
8
9     fclose(notas);
10    return 0;
11 }
```

Outras funções

- Existem ainda outras funções para manipulação de arquivos:
  - fgetc: lê um caractere do arquivo.
  - fgets: lê string até o caractere '\n'
  - fputc: escrever um caractere do arquivo.
  - fseek: posiciona a leitura dentro do arquivo (SEEK_SET, SEEK_CUR, SEEK_END).
  - entre outras...
- Verifique essas funções em : www.cplusplus.com
- Existem ainda as funções específicas para arquivos binários (serão apresentadas a seguir).

Resumindo

- O processo de utilização de um arquivo envolve, no mínimo, três etapas:
  - criação ou abertura do arquivo : **fopen**
  - leitura ou escrita de dados no arquivo: **fprintf** e **fscanf**
  - fechamento do arquivo: **fclose**

Arquivos Binários

- Variáveis **int** ou **float** tem tamanho fixo (por exemplo, 4 bytes) na memória.
- Representação em texto precisa de um número variável de dígitos (10, 5.673, 100.340).
- Armazenamento em arquivo análogo ao utilizado em memória permite reduzir o tamanho do arquivo e permite busca não sequencial.
- Esta é a motivação para o uso de arquivos binários!

Manipulando arquivos binários - Funções **fwrite** e **fread**

- O arquivo binário é também aberto pela função **fopen**.
- As funções **fwrite** e **fread** permitem a escrita e a leitura de dados.

Manipulando arquivos binários - Funções **fwrite** e **fread**

```c
int *le_vetor_texto(int *n){
    FILE *fr;
    int *v, i;
    fr = fopen ("v-in.txt", "r");
    if (fr == NULL) {
        perror("v-in.txt:");
        exit(-1);
    }
    fscanf(fr, "%d", n);
    v = (int *) malloc (*n * sizeof(
        int));
    for (i = 0; i < *n; i++)
        fscanf(fr, "%d", &v[i]);
    fclose(fr);
    return v;
}
```

```c
int *le_vetor_bin(int *n){
    FILE *fr;
    int *v;
    fr = fopen ("v.bin", "r");
    if (fr == NULL) {
        perror("v.bin: ");
        exit(-1);
    }
    fread(n, sizeof(int), 1, fr);
    v = (int *) malloc (*n * sizeof(
        int));
    fread(v, sizeof(int), *n, fr);
    fclose(fr);

    return v;
}
```

Manipulando arquivos binários - Funções **fwrite** e **fread**

```c
void escreve_vetor_texto(int *v, int n){
    FILE *fw;
    int i;
    fw = fopen ("v-out.txt", "w");
    if (fw == NULL){
        perror("v-out.txt ");
        exit(-1);
    }
    fprintf(fw, "%d\n", n);
    for (i = 0; i < n; i++)
        fprintf(fw, "%d\n", v[i]);
    fclose(fw);
}
```

```c
void escreve_vetor_bin(int *v, int n){
    FILE *fw;
    fw = fopen ("v.bin", "w");
    if (fw == NULL){
        perror("v.bin ");
        exit(-1);
    }
    fwrite(&n, sizeof(int), 1, fw);
    fwrite(v, sizeof(int), n, fw);
    fclose(fw);
}
```

```c
int main() {
    int *v1, *v2;
    int n1, n2;
    v1 = le_vetor_texto(&n1);
    escreve_vetor_bin(v1, n1);
    v2 = le_vetor_bin(&n2);
    escreve_vetor_texto(v2, n2);
    free(v1);
    free(v2);
    return 0;
}
```

Arquivos Binários e Registros

```c
typedef char Disc[5];
typedef struct reg{
     int RA;
     char nome[30];
     Disc matriculas[6];
}Reg_aluno;

#define N_REGS 2

Reg_aluno dados[N_REGS] = {
   {12436, "Maria", {"MC102", "MA141", "F 128", "F 129"}},
   {12232, "Joao", {"MC202", "MA211", "F 228", "F 229"}}
};
void escreve_dados(){
     FILE *fw;
     fw = fopen ("dados_aluno.bin", "w");
     if (fw == NULL){
          perror("dados_aluno.bin ");
          exit(-1);
     }
     fwrite(dados, sizeof(Reg_aluno), N_REGS, fw);
     fclose(fw);
}
int main(){
     escreve_dados();
     return 0;
}
```

Problemas: Trabalhar com texto e arquivo binário não é uma boa prática!

Resumindo

- O processo de utilização de um arquivo binário envolve, no mínimo, três etapas:
  - criação ou abertura do arquivo : **fopen**
  - leitura ou escrita de dados no arquivo: **fwrite** e **fread**
  - fechamento do arquivo: **fclose**

Exercícios

1. Faça uma função que retorne o número de linhas de um arquivo.
2. Faça uma função que retorne o número de caracteres de um arquivo. Espaços e quebras de linha não devem ser contados.
3. Faça um programa que leia de um arquivo as informações sobre o nome e as 2 notas de diversos alunos. Imprima a listagem de alunos e suas respectivas médias. O programa deve criar o arquivo de entrada e criar um arquivo de saída com o nome e a média de cada aluno. Armazene as informações dos alunos em um registro (struct).
4. Adicione ao programa anterior uma função para calcular o desvio padrão e a variância das médias dos alunos. Grave essa informação no final do arquivo de saída.

Na próxima aula...

Recursão