Algoritmos e Programação II

E/S com Arquivo

Sumário

- *Stream*
- Arquivo
- Sistema de Arquivos C ANSI

Stream

- *Stream* é um dispositivo lógico que representa um buffer no sistema de arquivos
  - C é projetado para vários dispositivos: terminais, acionadores de disco, acionadores de fita, dentre outros.

- *Streams* são independentes do dispositivo físico
  - A mesma função pode ser usada para escrever no disco ou em outro dispositivo de armazenamento permanente.

- *Streams* podem ser de Texto ou Binária

Stream

- *Stream* de Texto
  - É uma sequência de caracteres
  - C ANSI não exige uma *stream* de texto organizada em linhas determinadas por um caractere de nova linha (isso é opcional).
  - Não há uma relação um para um entre os caracteres escritos ou lidos e aqueles nos dispositivos externos.

- *Stream* Binária
  - É uma sequência de bytes com uma correspondência de um para um com aqueles encontrados no dispositivo externo (não ocorre tradução de caracteres).
  - O número de bytes escritos ou lidos é o mesmo do dispositivo externo.

Arquivo

- Pode ser um arquivo em disco, uma impressora ou um terminal, dentre outros.

- É realizada uma associação entre *stream* e arquivo através da **operação de abertura**
  - Pode acontecer leitura e/ou escrita, quanto o arquivo estiver aberto.

Arquivo (2)

- A associação entre Arquivo e Stream é desfeita com a **operação de fechamento**
  - Se o programa é encerrado inesperadamente (“abortado”), os arquivos não são fechados, consequentemente as tabelas de arquivos no sistema ficam desatualizadas, possibilitando a ocorrência de erros.

- Cada *Stream* associada com um arquivo tem uma estrutura de controle de arquivo do tipo `FILE`
  - Usar o cabeçalho: #include<stdio.h>

Ponteiro de Arquivo

- Um **Ponteiro de Arquivo**
  - é um ponteiro para o nome, status e posição atual no arquivo.
  - Um ponteiro de arquivo define um arquivo em disco.

- É uma variável ponteiro do tipo FILE

```
FILE *fp;
```

Abertura de Arquivo

- Antes de manipularmos um arquivo, é necessário **abrir o arquivo.**

```
FILE *fp; //fp é a variável ponteiro para o arquivo

//para abrir o arquivo, usamos o comando fopen
fp = fopen(“nome.ext”, “modo de abertura”);
```

Abertura de Arquivo

- Antes de manipularmos um arquivo, é necessário **abrir o arquivo.**

```
FILE *fp; //fp é a variável ponteiro para o arquivo

//para abrir o arquivo, usamos o comando fopen
fp = fopen("nome.ext", "modo de abertura");
```

- Nome do arquivo com a extensão.
- Caminho ou localização do arquivo pode estar incluído.
- Pode ser usado por meio de uma variável.

- Modo de abertura do arquivo indica o que poderemos fazer com o arquivo (ler, escrever, criar novo...).
- Veremos nos próximos slides sequência são as possibilidades.

Abertura de Arquivo

```
FILE *fp; //fp é a variável ponteiro para o arquivo

//o comando fopen pode conter uma variável no lugar do
//nome do arquivo
char var[30];
…
fp = fopen(var, “modo de abertura”);
```

Modo de abertura de arquivos

Modo	Significado
r	Abre um arquivo texto para leitura.
w	Cria um arquivo texto para escrita. Se já existe,este é apagado e criado um novo.
a	Anexa a um arquivo texto. Se já existe, não é apagado.
rb	Abre um arquivo binário para leitura.
wb	Cria um arquivo binário para escrita. Se já existe,este é apagado e criado um novo.
ab	Anexa a um arquivo binário.
r+	Abre um arquivo texto para leitura/escrita.
w+	Cria um arquivo texto para leitura/escrita.
a+	Anexa ou cria um arquivo texto para leitura/escrita.
r+b	Abre um arquivo binário para leitura/escrita.
w+b	Cria um arquivo binário para leitura/escrita.
a+b	Anexa a um arquivo binário para leitura/escrita.

Fechar arquivo

- **Fechando um arquivo**

```
fclose(fp);
```

- Retorna ZERO, se houve sucesso na operação.

- Ao ser executada a função, qualquer dado que esteja no buffer do disco é gravado no arquivo e depois o arquivo é fechado.

Escrevendo e lendo um caractere

Para escrever ou ler um caractere de um arquivo, ele deve estar aberto

```
…
char a = 'A';
putc(a, fp); //ou fputc(a, fp);
a = getc(fp); //ou a = fgetc(fp);
…
```

- `putc()` ou `fputc()` devolvem *End Of File* (EOF) se houver erro. Caso contrário, devolvem o caractere escrito.

- `getc()` ou `fgetc()` devolvem EOF quando o final do arquivo for atingido ou quando ocorrer um erro.

Teste o Exemplo 1 a seguir...

Identifique a finalidade de cada comando associado ao uso do arquivo.

Exemplo 1 – Escrita de caracteres em arquivo

```
#include<stdio.h>

main(){
  FILE *fp;
  char nome[15], ch;

  printf("Digite o nome do arquivo a ser criado: ");
  gets(nome);

  if((fp = fopen(nome, "w"))==NULL){
      printf("Arquivo não pode ser aberto...PRESSIONE QUALQUER TECLA PARA CONTINUAR");
      getchar();
      exit(1);
  }
  printf("ARQUIVO TEXTO ABERTO COM SUCESSO!!!\n Digite seu texto(# - termina): \n");
  do{
    ch = getchar();
    if (ch!='#') putc(ch, fp);
  } while (ch!='#');

  printf("\nTérmino da Digitação");
  fclose(fp);
}
```

- Localize em seu diretório (pasta) corrente o arquivo criado, conforme o nome que você digitou.

- Abra o arquivo para visualizar seu conteúdo (o arquivo pode ser aberto, por exemplo, com o Bloco de Notas, pois foi criado como um arquivo texto - foi usada a diretiva w).

- Como o arquivo criado possui stream de texto, é possível visualizar seu conteúdo

Para **testar o Exemplo 2** a seguir, informe o mesmo nome de arquivo do Exemplo 1.

Identifique a finalidade de cada comando associado ao uso do arquivo.

Exemplo 2 – Leitura caracteres em arquivo

```
#include<stdio.h>
main(){
  FILE *fp;
  char nome[15], ch;

  printf("Digite o nome do arquivo a ser aberto: ");
  gets(nome);

  if((fp = fopen(nome, "r"))==NULL){
      printf("Arquivo não pode ser aberto...PRESSIONE QUALQUER TECLA PARA CONTINUAR");
      getchar();
      exit(1);
  }
  printf("ARQUIVO TEXTO ABERTO COM SUCESSO!!!\n");

  ch = getc(fp);
  while (ch!=EOF){
    putchar(ch);
    ch = getc(fp);
  }
  fclose(fp);
}
```

Arquivo Binário

- Um arquivo binário deve ser manipulado com a diretiva apropriada, conforme a tabela do slide 11.
- Quando está sendo manipulado um arquivo binário, o fim do arquivo, na leitura dos dados binários, é indicado pelo EOF.
- A função `feof()` retorna verdadeiro se o final do arquivo foi atingido; caso contrário, retorna ZERO

```
FILE *fp;
while(!feof(fp)) ch=getc(fp);
```

O **Exemplo 3** faz a cópia de um arquivo existente, ou seja, cria um novo arquivo com o conteúdo de um arquivo existente.

Verifique ao executar e analisar o código...

Lembre-se que neste exemplo, o arquivo criado chama-se **`copia.txt`**

Exemplo 3 – Cópia de 2 arquivos

```
#include<stdio.h>

int main(){
  FILE *forigem, *fdestino;
  char origem[30], ch;

  printf("Digite o nome do arquivo a ser copiado: ");
  gets(origem);

  if((forigem = fopen(origem, "r"))==NULL){
      printf("Arquivo origem não pode ser aberto...PRESSIONE QUALQUER TECLA PARA CONTINUAR");
      getchar();
      exit(1);
  }
  printf("ARQUIVO %s ABERTO COM SUCESSO!!!\n", origem);

  if((fdestino = fopen("copia.txt", "w"))==NULL){
      printf("Arquivo destino não pode ser aberto...PRESSIONE QUALQUER TECLA PARA
   CONTINUAR");
      getchar();
      exit(1);
  }

  //continua no próximo slide...
```

Exemplo 3 – Cópia de 2 arquivos

```
while (!feof(forigem)){
    ch = getc(forigem);
    if (!feof(forigem))
        putc(ch, fdestino);
  }
  printf("ARQUIVO \"copia.txt\" CRIADO COM SUCESSO!!!\n");

  getchar();
  fclose(forigem);
  fclose(fdestino);

  return 0;
}
```

Lembre-se de visualizar o conteúdo dos dois arquivos em seu diretório (pasta) corrente.

Sistema de Arquivos C ANSI

- **Posicionando o indicador de posição no início do arquivo**

```
rewind(fp)
```

- O arquivo deve ter sido aberto no modo leitura ou leitura/escrita

Crie um novo programa, modificando o exemplo 3, para mostrar o arquivo copiado.
Use a função **rewind** para mostrar o arquivo desde o início!

Sistema de Arquivos C ANSI

- **Determinando erros nas operações com arquivos**

```
ferror(fp)
```

`ferror()` retorna verdadeiro se ocorreu erro na última operação com o arquivo

- Deve ser chamada após cada operação com arquivo.

Sistema de Arquivos C ANSI

- **Removendo arquivos**

  `remove(nome_arq)`

  remove() retorna ZERO se houve sucesso na operação
- Teste o exemplo...

```
#include<stdio.h>

int main(){
  char origem[30];

  printf("Digite o nome do arquivo a ser apagado: ");
  gets(origem);

  remove(origem);

  return 0;
}
```

Escrevendo e lendo blocos de qualquer tipo de dados

- **O arquivo deve estar aberto para dados binários**.
- As funções **fread()** e **fwrite()** podem ler e escrever blocos de qualquer tipo de dado
  - `fread()` devolve o número de itens lidos
  - `fwrite()` devolve o número de itens escritos

Escrevendo e lendo blocos de qualquer tipo de dados

- As funções **fread()** e **fwrite()** são utilizadas para ler/escrever estruturas (*struct*) definidas pelo usuário.

- Em ambas funções deve ser especificado, nesta ordem:
    - Endereço da variável que irá receber o dado lido ou que contém o dado a ser gravado.
    - O número de bytes que serão lidos/gravados.
    - O número de itens escritos.
    - O ponteiro para o arquivo.

Teste o Exemplo 4 a seguir...

Observe que a arquivo usado é binário e as funções para ler e escrever são fread() e fwrite(), respectivamente.

Exemplo 4 – fread() e fwrite()

```
#include<stdio.h>

int main(){
  FILE *fp;
  char arq[15];
  int idade;
  float salario;

  printf("Digite o nome do arquivo a ser criado: ");
  gets(arq);

  if((fp = fopen(arq, "w+b"))==NULL){
      printf("Arquivo não pode ser aberto...PRESSIONE QUALQUER TECLA PARA CONTINUAR");
      getchar();
      exit(1);
  }

  printf("Idade: \n");    scanf("%d", &idade);
  printf("Salario: \n");  scanf("%f", &salario);

//Continua no próximo slide…
```

Exemplo 4 – fread() e fwrite()

```
    fwrite(&idade, sizeof(int), 1, fp);
    fwrite(&salario, sizeof(float), 1, fp);

    idade=0; salario=0.0;
    rewind(fp);

    fread(&idade, sizeof(int), 1, fp);
    fread(&salario, sizeof(float), 1, fp);

    printf("Idade %d - Salário %f\n", idade, salario);
    fclose(fp);

    return 0;
}
```

Algumas Funções do Sistema de Arquivos C ANSI

Função	**Finalidade**
fprintf()	Printf para arquivo
scanf()	Scanf de arquivo
feof()	**Verdadeiro** se o final do arquivo é atingido
ferror()	**Verdadeiro** se ocorreu um erro
rewind()	Indicador de posição, vai para o início do arquivo
remove()	Apaga um arguivo
fflush()	Descarrega um arquivo

Algumas Funções do Sistema de Arquivos C ANSI

Função	Finalidade
fopen()	Abre um arquivo
fclose()	Fecha um arquivo
putc()	Escreve um caractere em um arquivo
fputc()	Equivalente à putc()
getc()	Lê um caractere de um arquivo
fgetc()	Equivalente à getc()
fseek()	Posiciona o arquivo em um byte específico
fgets()	Lê uma linha inteira de um arquivo
fputs()	Grava uma linha inteira em um arquivo

Bibliografia

SCHILDT, Herbert. *C completo e total.* São Paulo: Makron Books, 1997.

EVARISTO, Jaime. *Aprendendo a programar programando na linguagem C.* Maceió: Vivali, 2007.