Disciplina:

# Programação Computacional

Prof. Fernando Rodrigues
e-mail: fernandorodrigues@sobral.ufc.br

## Aula 14.2: Programação em C

- Funções Avançadas de manipulação de arquivos.

# Forçando a escrita dos dados do buffer – fflush() (Exemplo)

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  #include <string.h>
04  int main(){
05    FILE *arq;
06    char string[100];
07    int i;
08    arq = fopen("arquivo.txt","w");
09    if(arq == NULL){
10      printf("Erro na abertura do arquivo");
11      system("pause");
12      exit(1);
13    }
14    printf("Entre com a string a ser gravada no arquivo:");
15    gets(string);
16    for(i = 0; i < strlen(string); i++)
17      fputc(string[i], arq);
18
19    fflush(arq);
20    fclose(arq);
21    system("pause");
22    return 0;
23  }
```

# Gravando uma string e o seu tamanho

**Exemplo: gravando uma string e o seu tamanho**

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  #include <string.h>
04  int main(){
05    FILE *arq;
06    arq = fopen("ArqGrav.txt","wb");
07    if(arq == NULL){
08      printf("Erro\n");
09      system("pause");
10      exit(1);
11    }
12    char str[20] = "Hello World!";
13    int t = strlen(str);
14    fwrite(&t,sizeof(int),1,arq);
15    fwrite(str,sizeof(char),t,arq);
16    fclose(arq);
17    system("pause");
18    return 0;
19  }
```

# Lendo uma string e o seu tamanho

**Exemplo: lendo uma string e o seu tamanho**

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    FILE *arq;
05    arq = fopen("ArqGrav.txt","rb");
06    if(arq == NULL){
07      printf("Erro\n");
08      system("pause");
09      exit(1);
10    }
11    char str[20];
12    int t;
13    fread(&t,sizeof(int),1,arq);
14    fread(str,sizeof(char),t,arq);
15    str[t] = '\0';
16    printf("%s\n",str);
17    fclose(arq);
18    system("pause");
19    return 0;
20  }
```

# Gravando uma matriz usando fprintf()

**Exemplo: gravando uma matriz**

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    FILE *arq;
05    arq = fopen("matriz.txt","w");
06    if(arq == NULL){
07      printf("Erro\n");
08      system("pause");
09      exit(1);
10    }
11    int mat[2][2] = {{1,2},{3,4}};
12    int i,j;
13    for(i = 0; i < 2; i++)
14      for(j = 0; j < 2; j++)
15        fprintf(arq,"%d\n",mat[i][j]);
16    fclose(arq);
17    system("pause");
18    return 0;
19  }
```

# Lendo uma matriz: feof() e fscanf()

**Exemplo: lendo uma matriz**

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    FILE *arq;
05    arq = fopen("matriz.txt","r");
06    if(arq == NULL){
07      printf("Erro\n");
08      system("pause");
09      exit(1);
10    }
11    int v,soma=0;
12    while(!feof(arq)){
13      fscanf(arq,"%d",&v);
14      soma += v;
15    }
16    printf("Soma = %d\n",soma);
17    fclose(arq);
18    system("pause");
19    return 0;
20  }
```

# Fim do arquivo - feof()

- A constante EOF (“End Of File”) indica o fim de um arquivo.
- Porém, quando manipulando dados binários, um valor inteiro igual ao valor  da constante EOF pode ser lido.
- Nesse caso, se utilizarmos a constante EOF para verificar se chegamos ao final do arquivo, vamos receber a confirmação quando, na verdade, isso ainda não aconteceu.
- Para evitar esse tipo de situação, a linguagem C inclui a função feof(), que determina quando o final de um arquivo foi atingido. Seu protótipo é:
  - int feof(FILE *fp)
- Basicamente, a função feof() recebe como parâmetro o ponteiro fp, que determina o arquivo a ser verificado. Como resultado, a função retorna um valor inteiro igual a ZERO se ainda não tiver atingido o final do arquivo.
- Um valor de retorno diferente de zero significa que o final do arquivo já foi atingido.

# Erro ao acessar um arquivo - ferror()

- Ao se trabalhar com arquivos, diversos tipos de erros podem ocorrer: um comando de leitura pode falhar, pode não haver espaço suficiente em disco para gravar o arquivo etc.
- Para determinar se uma operação realizada com o arquivo produziu algum erro, existe a função ferror(), cujo protótipo é:
  - int ferror(FILE *fp)
- Basicamente, a função ferror() recebe como parâmetro o ponteiro fp, que determina o arquivo que se quer verificar.
- A função verifica se o indicador de erro associado ao arquivo está marcado e retorna um valor igual a zero se nenhum erro ocorreu. Do contrário, a função retorna um número diferente de zero.
- Como cada operação modifica a condição de erro do arquivo, a função ferror() deve ser chamada logo após cada operação realizada com o arquivo.

# Movendo-se dentro do arquivo - fseek()

- De modo geral, o acesso a um arquivo é quase sempre feito de modo sequencial.
- Porém, a linguagem C permite realizar operações de leitura e escrita randômica. Para isso, usa-se a função fseek(), cujo protótipo é:
  - int fseek(FILE *fp, long numbytes, int origem)
- Basicamente, a função fseek() move a posição atual de leitura ou escrita no arquivo para um byte específico, a partir de um ponto especificado.
- A função fseek() recebe três parâmetros de entrada:
  - **fp**: o ponteiro para o arquivo em que se deseja trabalhar.
  - **numbytes**: é o total de bytes a partir de “origem” a ser pulado.
  - **origem**: determina a partir de onde os “numbytes” de

O valor do parâmetro **numbytes** pode ser negativo, dependendo do tipo de movimentação que formos realizar.

# Usado a função fseek()

**Exemplo: usando a função fseek()**

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  struct cadastro{ char nome[20], rua[20]; int idade;};
04  int main(){
05    FILE *f = fopen("arquivo.txt","wb");
06    if(f == NULL){
07       printf("Erro na abertura\n");
08       system("pause");
09       exit(1);
10    }
11    struct cadastro c,cad[4] = {"Ricardo","Rua 1",31,
12                                "Carlos","Rua 2",28,
13                                "Ana","Rua 3",45,
14                                "Bianca","Rua 4",32};
15    fwrite(cad,sizeof(struct cadastro),4,f);
16    fclose(f);
17    f = fopen("arquivo.txt","rb");
18    if(f == NULL){
19       printf("Erro na abertura\n");
20       system("pause");
21       exit(1);
22    }
23    fseek(f,2*sizeof(struct cadastro),SEEK _ SET);
24    fread(&c,sizeof(struct cadastro),1,f);
25    printf("%s\n%s\n%d\n",c.nome,c.rua,c.idade);
26    fclose(f);
27    system("pause");
28    return 0;
29  }
```

# Valores para a função fseek()

- A função fseek() retorna um valor inteiro igual a ZERO quando a movimentação dentro do arquivo for bem-sucedida. Um valor de retorno diferente de zero significa que houve um erro durante a movimentação.
- Os valores possíveis para o parâmetro origem são definidos por constante na biblioteca stdio.h e são:

| Constante | Valor | Significado |
|---|---|---|
| SEEK_SET | 0 | Início do arquivo |
| SEEK_CUR | 1 | Ponto atual no arquivo |
| SEEK_END | 2 | Fim do arquivo |

- Portanto, para movermos numbytes a partir do início do arquivo, a origem deve ser SEEK_SET. Se quisermos mover a partir da posição atual em que estamos no arquivo, devemos usar a constante SEEK_CUR. Por fim, se quisermos mover a partir do final do arquivo, a constante SEEK_END deverá ser usada.

# Voltando ao começo do arquivo - rewind()

- Outra opção de movimentação dentro do arquivo é simplesmente retornar para o seu início. Para tanto, usa-se a função rewind(), cujo protótipo é:

  - void rewind(FILE *fp)

- A função rewind() recebe como parâmetro de entrada apenas o ponteiro para o arquivo que se deseja retornar para o início.

# Excluindo um arquivo - remove()

- Além de permitir manipular arquivos, a linguagem C também permite excluí-los do disco rígido. Isso pode ser feito facilmente utilizando a função remove(), cujo protótipo é:
  - int remove(char *nome_do_arquivo)

- Diferentemente das funções vistas até aqui, a função remove() recebe como parâmetro de entrada o caminho e o nome do arquivo a ser excluído do disco rígido, e não um ponteiro para FILE.

- Como resultado, essa função retorna um valor inteiro igual a ZERO quando houver sucesso na exclusão do arquivo. Um valor de retorno diferente de zero significa que houve um erro durante a sua exclusão.

# Renomeando um arquivo - rename()

- A função rename(),  cujo protótipo é:
  - int rename(const char *,const char *)
- É útil quando se precisa renomear um arquivo.

- Esta função recebe como parâmetros de entrada o nome atual do arquivo a ser renomeado e o novo nome a ser dado a tal arquivo.

- Como resultado, essa função retorna um valor inteiro igual a ZERO quando houver sucesso na alteração do nome do arquivo. Um valor de retorno diferente de zero indica que houve um erro durante a renomeação.

# Sabendo a posição atual dentro do arquivo - ftell()

- Outra operação bastante comum é saber onde estamos dentro de um arquivo.
- Para realizar essa tarefa, usamos a função ftell(), cujo protótipo é:
  - long int ftell(FILE *fp)
- Basicamente, a função ftell() recebe como parâmetro o ponteiro fp, que determina o arquivo a ser manipulado. Como resultado, a função ftell() retorna a posição atual dentro do fluxo de dados do arquivo:
  - Para arquivos binários, o valor retornado indica o número de bytes lidos a partir do início do arquivo.
  - Para arquivos texto, não existe garantia de que o valor retornado seja o número exato de bytes lidos a partir do início do arquivo.
  - Se ocorrer um erro, o valor −1 no formato long é retornado.

# ftell() - Exemplo: descobrindo o tamanho de um arquivo

**Exemplo: descobrindo o tamanho de um arquivo**

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  #include <string.h>
04  int main(){
05    FILE *arq;
06    arq = fopen("arquivo.bin","rb");
07    if(arq == NULL){
08      printf("Erro na abertura do arquivo");
09      system("pause");
10      exit(1);
11    }
12    int tamanho;
13    fseek(arq,0,SEEK _ END);
14    tamanho = ftell(arq);
15    fclose(arq);
16    printf("Tamanho do arquivo em bytes: %d:",tamanho);
17    system("pause");
18    return 0;
19  }
```

# Exercícios II

1) Escreva um programa que receba via linha de comando dois nomes de arquivos: um atual (existente) e um outro como sendo o novo nome do arquivo. Em seguida, renomeie o arquivo de acordo com os parâmetros passados.

2) Escreva um programa que receba via linha de comando 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). O nome a ser dado ao novo arquivo deve ser formado pela concatenação dos nomes dos dois arquivos dados que devem ser ligados por um “underline” ( _ ), retirando-se a extensão dos mesmos e incluindo a extensão “.txt” ao final.
P.Ex: Caso os nomes dos arquivos sejam: ArqTexto1.doc e ArqTexto2.rtf, o nome do novo arquivo criado deve ser: “ArqTexto1_ArqTexto2.txt”

3) Crie um programa que defina três constantes: N, M e A (via diretivas de compilação), onde N é o número de valores que deverão ser gerados aleatoriamente no intervalo de 0 até M e A é o nome de um arquivo (que deverá ter formato binário) e que armazenará um vetor com os N valores inteiros aleatórios que serão gerados.

# FIM