Aula 14 – Oficina de Programação
Tópicos Especiais em C: Arquivos

Profa. Elaine Faria

UFU - 2017

Uso da Memória Secundária

- Em muitos casos necessitamos da memória secundária (auxiliar), para armazenar informações
- Podemos utilizar discos como HD, Cds, Dvds, etc.
- Além disso, podemos utilizar **arquivos** para armazenar dados
- Um **arquivo** é uma coleção de bytes referenciados por um nome único

Arquivos

- Entende-se por arquivo as estruturas de dados armazenadas na memória secundária (como discos) que posteriormente podem ser lidas e alteradas
- Não há perda de dados ao se desligar o computador e/ou programa
- A biblioteca utilizada para manipulação de arquivos é **<stdio.h>**
- Uma vez que o arquivo está aberto, informações podem ser trocadas entre ele e o programa

Algumas funções para manipulação de arquivos

Nome	Função
fopen()	Abre um arquivo
fclose()	Fecha um arquivo
getc()	Lê um caractere de um arquivo
fgetc()	O mesmo que getc()
fseek()	Posiciona o arquivo em um byte específico padrão
fprintf()	É para um arquivo o que o printf() é para o console
fscanf()	É para um arquivo o que o scanf() é para o console
fread()	Lê um registro do arquivo
fwrite()	Grava um registro no arquivo
feof()	Retorna verdadeiro se o fim do arquivo for atingido
ferror()	Retorna verdadeiro se ocorreu um erro
rewind()	Realoca o indicador de posição de arquivo no início do arquivo
remove()	Apaga um arquivo
fflush()	Descarrega um arquivo

Formato de um arquivo

- Arquivos podem ser classificados em dois tipos:
  → Arquivos de texto (ou formatados)
  → Arquivos binários (ou não-formatados)

Formato de um arquivo

→ Arquivos de texto (ou formatados)

- Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de texto simples
- Os dados são gravados como caracteres de 8 bits

Formato de um arquivo

→ Arquivos binários (ou não-formatados)

- Os dados são gravados na forma binária (do mesmo modo que estão na memória).

Leitura e gravação em disco

- Existem 4 diferentes formas de acessar arquivos:

  1) Dados são lidos e escritos um caracter por vez

  2) Dados são lidos e escritos como “strings”

  3) Dados são lidos e escritos de modo formatado

  4) Dados são lidos e escritos em um formato chamado registro ou bloco

Ponteiros para arquivos

- Para declarar um ponteiro para arquivo é utilizado o tipo de dados FILE
- Esse ponteiro serve para identificar um arquivo no disco, permitindo a gravação, alteração exclusão de dados de um arquivo
- Declaração de um ponteiro para arquivo:

```
FILE *fp;
```

Manipulação de Arquivos

- Três etapas básicas

  1) Abrir o arquivo;

  2) Ler e/ou gravar os dados desejados;

  3) Fechar o arquivo.

Função fopen( )

- Função para abrir o arquivo
- O ponteiro recebe o endereço de memória ocupado pelo arquivo, sendo que, se houver erro o ponteiro valerá nulo (NULL).

```
FILE * fp;
fp = fopen("arquivo.txt", "w");
if (fp == NULL )
{
        printf("Não foi possível abrir o arquivo.\n");
        exit(1);  // força o término da execução da rotina
}
```

Função fopen()

```
FILE *fp;
fp = fopen("arquivo.txt", "w");
```

- Para gerar um código de programa que abre um arquivo, o compilador precisa conhecer 3 coisas:

1- O **nome** do arquivo
2- O **tipo** de abertura
3- **Onde guardar** informações sobre o arquivo

- Tipos de abertura de arquivo:

"r" para leitura
"w" para gravação
"a" para adicionar dados

Função fopen()

- Dois modificadores podem ser usados junto ao tipo:
  - Letra “b” para modo binário
  - Sinal “+” quando o arquivo é aberto para escrita e leitura.

- Lista completa de opções de tipo para fopen():

Tipo	Significado
“r”	Abre um arquivo texto somente para leitura. O arquivo deve existir.
“r+”	Abre um arquivo texto para leitura e gravação. O arquivo deve existir.

Tipo	Significado
“w”	Abre o arquivo texto somente para escrita a partir do início. Apagará o arquivo se ele já existir, criará um novo se não existir.
“w+”	Abre o arquivo texto para leitura e gravação, apagando o conteúdo pré-existente. Se não existir, o arquivo será criado.
“a”	Abre um arquivo texto para escrita a partir do final. Não apaga o conteúdo pré-existente. Se o arquivo não existir um novo será criado.
“a+”	Abre um arquivo texto para leitura e gravação. A gravação adiciona dados ao fim do arquivo existente ou um novo arquivo será criado.
“rb”	Abre um arquivo binário somente para leitura. O arquivo deve estar presente no disco.
“wb”	Abre um arquivo binário somente para gravação. Se o arquivo estiver presente ele será destruído e reinicializado. Se não existir, ele será criado.

Tipo	Significado
“ab”	Abre um arquivo binário somente para gravação. Os dados serão adicionados ao fim do arquivo existente, ou um novo arquivo será criado.
“rb+”	Abre um arquivo binário para leitura e gravação. O arquivo deve existir e pode ser atualizado.
“wb+”	Abre um arquivo binário para leitura e gravação. Se o arquivo existir ele será destruído e reinicializado. Se não existir, será criado.
“ab+”	Abre um arquivo binário para leitura e gravação. A gravação adicionará dados ao fim do arquivo existente ou um novo arquivo será criado.

Cuidado ao abrir arquivos

```
fp= fopen("arquivo.txt", "w");

if (fp==NULL)
    printf("\nHouve problemas, verifique.\n");
else
    printf("\nArquivo aberto corretamente.\n");
```

- Problemas:

  Gravação: não tem espaço em disco

  Leitura: o arquivo não existe

- **Sempre** verifique se o arquivo foi aberto com sucesso, antes de escrever ou ler.

Função fclose()

- Fecha um arquivo, utilizando-se do ponteiro.
- A função retorna 0 (zero) se não houver problemas, caso contrário, retorna um erro. Exemplo:

```
int main()
{
    FILE *fp;
    int erro;

    fp= fopen("arquivo.dat", "w");

    erro= fclose(fp);

    if (erro==0)
        printf("\nArquivo fechado com sucesso.\n");
    else
       printf("\nErro no fechamento.\n");

    return 0;
}
```

Leitura e gravação em disco

- Formas de acessar arquivos:

1) Dados são lidos e escritos um caracter por vez

→ Funções:

- fgetc() e fputc()

Função fputc ( )

- → Grava **um caracter** no arquivo

```
int fputc(char ch, FILE *fp)
```

ch é o caracter a ser gravado

fp é o ponteiro devolvido por fopen

Exemplo:

```
FILE *fp;
fp = fopen("arquivo.txt", "w");
fputc('a', fp);
```

```
int main ()
{
    FILE * pFile;
    char string[100];
    int c;
     pFile=fopen("alfabeto.txt","w+");

     if (pFile == NULL) {
       printf("\nProblemas na abertura do arquivo!.\n");
       exit(1);
     }
     printf(“Entre com a string a ser gravada no arquivo”);
     gets(string);
     for (c = 0 ; c < strlen(string) ; c++) {
        fputc (string[c] , pFile);
     }
     fclose (pFile);
     return 0;
}
```

Função fgetc ( )

→ Lê **um caracter** do arquivo

```
char fgetc(FILE *fp)
```

fp é o ponteiro devolvido por fopen

Exemplo:

```
FILE *fp;
fp = fopen("teste.txt", "r");
ch = fgetc(fp);
while (ch != EOF)
{
  printf("%c", ch);
  ch = fgetc(fp);
}
```

```
int main ()
{
  FILE * pFile;
  char c;

  pFile=fopen("alfabeto.txt","r");

  if (pFile == NULL){
     printf("\nProblemas na abertura do arquivo!.\n");
     exit(1);
   }

  c = fgetc(pFile);
  while(c!=EOF){
     printf("%c", c);
     c = fgetc(pFile);
  }

  fclose (pFile);
  return 0;
}
```

Fim de arquivo (EOF)

- Definida em **<stdio.h>**
- Retornada por **fgetc()** quando tenta ler além do final de um arquivo
- Indica que o final do arquivo foi atingido
- Não pode ser utilizada com arquivos binários
- Pode ser utilizada com arquivos de texto

Leitura e gravação em disco

- Formas de acessar arquivos:

  2) Dados são lidos e escritos como "strings"

  → Funções: fgets() e fputs()

Função fputs ( )

→ Grava **uma string** no arquivo

```
int fputs(char *s, FILE *fp)
```

`s` é a string a ser gravada

`fp` é o ponteiro devolvido por fopen

→ Retorna um valor positivo quando a escrita é bem sucedida; caso contrário, retorna EOF

Função fputs ( )

```
int main ()
{
    FILE * pFile;
    char string [100];

    printf ("Escreva uma frase: ");
    gets (string);

    pFile = fopen ("frase.txt","w+");

    fputs (string,pFile);

    fclose (pFile);
    return 0;

}
```

Função fgets ( )

→ Lê **uma linha** por vez do arquivo

```
char *fgets(char *s, int n, FILE *fp)
```

s é a string a ser lida

n número máximo de caracteres que serão lidos no vetor **s**

fp é o ponteiro devolvido por fopen

Função fgets ( )

- Lê caracteres até atingir um caractere `'\n'`, ou o final do arquivo ou o número máximo de caracteres especificado
- Escreve um caractere nulo `'\0'` após o último caractere armazenado no vetor
- Quando o final do arquivo é atingido antes de armazenar algum caractere no vetor, ela retorna **NULL**; caso contrário, retorna o argumento **`s`**

Função fgets ( )

```
int main()
{
   FILE * pFile;
   char string[100];
   char * presult;
   pFile = fopen ("frase.txt","r");
   presult = fgets(string, 100, pFile);
   while(presult != NULL) {
        printf("%s", string);
        presult = fgets(string, 100, pFile);
    }
   fclose (pFile);
   return 0;
}
```

Leitura e gravação em disco

- Formas de acessar arquivos:

  3) Dados são lidos e escritos de modo formatado

  → Funções: fscanf() e fprintf()

Função fprintf ( )

→ Grava um arquivo de modo **formatado**

```
fprintf(FILE *fp, char *s, … )
```

fp é o ponteiro devolvido por fopen

s string formatada

. . . variáveis

Função fprintf ( )

```
int main()
{
   FILE * pFile;
   int idade;
   char nome [100];

   pFile = fopen ("formatado.txt","w"); // Testar arquivo

    printf ("Digite seu nome: ");
    gets (nome);
    printf ("Digite sua idade:");
    scanf("%d", &idade);
    fprintf (pFile, "\nNome: %s, \nIdade: %d\n\n",nome, idade);

    fclose (pFile);
    return 0;
}
```

Função fscanf ( )

→ Lê dados **formatados**

```
fscanf(FILE *fp, char *s, … )
```

fp é o ponteiro devolvido por fopen

s string formatada

. . . endereço das variáveis

Função fscanf ( )

```
int main()
{
   char str [80];
   float f;
   FILE * pFile;

   pFile = fopen ("arq.txt","w+");
   fprintf (pFile, "%f %s", 3.1416, "PI");
   rewind (pFile);
   fscanf (pFile, "%f", &f);
   fscanf (pFile, "%s", str);
   fclose (pFile);
   printf ("Leitura do arquivo: %f e %s \n",f,str);
   return 0;
}
```

Leitura e gravação em disco

- Formas de acessar arquivos:

  4) Dados são lidos e escritos em um formato chamado registro ou bloco

  → Funções: fread() e fwrite()

  → Operações para leitura/escrita em arquivos binários

  → São usados para leitura e escrita, de sequências de bytes

Função fwrite( )

- Lê um array em memória e escreve no arquivo

int fwrite(void *mem, int n_bytes, int cont, FILE *fp);

→fwrite() toma 4 argumentos

1º : ponteiro para a localização da memória do dado a ser gravado.

2º : tamanho da representação binária a ser gravada.

3º : número inteiro que informa a fwrite() quantos itens do mesmo tipo serão gravados.

4º : ponteiro que controla o arquivo aberto para escrita.

```
int main()
{
        struct pessoa pes;
        FILE *fp;
        int erro, op;
        fp= fopen("teste.txt","wb");
        if (fp==NULL) exit(1);

        printf("\nNome: "); gets(pes.nome);
        printf("\nIdade:" ); scanf("%d",&pes.idade);
        printf("\nPeso:" ); scanf("%f",&pes.peso);
        fwrite(&pes, sizeof(struct pessoa), 1, fp);

        erro=fclose(fp);
        return 0;
}
```

```
struct pessoa
{
    char nome[30];
    int idade;
    float peso;
};
```

Função fread( )

- Lê dados em arquivo

int fread(void *mem, int n_bytes, int cont, FILE *fp);

→fread() possui 4 argumentos

1º : ponteiro para a localização da memória onde serão armazenados os dados lidos.

2º : indica a quantidade de bytes do tipo de dados a ser lido.

3º : quantidade de itens a serem lidos a cada chamada

4º : ponteiro que controla o arquivo aberto para escrita.

Função fread( )

→ Retorno da função fread():

- Número de itens lidos
- Deve ser o mesmo valor do terceiro argumento
- Quando é 0, a interpretação é ambígua
  - Pode indicar que o final do arquivo foi atingido
  - Pode indicar que ocorreu algum erro antes da leitura de algum elemento

```
int main()
{
        struct pessoa pes;
        FILE *fp;
        int erro, op;
        fp= fopen("teste.txt","rb");
        if (fp==NULL) exit(1);
       while(fread(&pes,sizeof(struct pessoa),1,fp)==1)
       {
            printf("\nNome: %s", pes.nome);
            printf("\nIdade: %d", pes.idade);
            printf("\nPeso: %0.2f\n", pes.peso);
        }
        erro=fclose(fp);
       return 0;
}
```

```
struct pessoa
{
    char nome[30];
    int idade;
    float peso;
};
```

Exercícios

1) Elabore um programa em C que leia os seguintes campos para o cadastro de um fornecedor:

- Nome
- Endereço
- Telefone
- E-mail

Grave estes dados no arquivo “forn.txt”

2) Elabore um programa em C que apresente na tela todos os fornecedores armazenados no arquivo “forn.txt”.