RAFAEL VIEIRA COELHO

ARQUIVOS

PROGRAMAÇÃO 2

CAPÍTULO 20: ARQUIVOS

https://books.goalkicker.com/CBook/

Arquivos em C

- Funções e tipos para manipulação de arquivos também são definidos em stdio.h;

- C define o tipo (não-primitivo) FILE.

- Uma variável FILE é um descritor de arquivo (ou fluxo de arquivo);

- As funções de manipulação trabalham com ponteiros para FILE;

- Há funções para abrir, fechar, escrever, ler e se movimentar em arquivos;

- Portanto, para trabalhar com arquivos, precisamos usar:

```
FILE *arquivo;
```

Abrindo um Arquivo

•A função fopen() abre um arquivo:

FILE *fopen(char *nomeArquivo, char *modo);

•nomeArquivo contém o nome ou caminho completo para o arquivo a ser aberto;

•modo define o tipo de uso que vai se fazer do arquivo.

•fopen() retorna o ponteiro para o descritor do arquivo aberto (FILE), ou nulo (NULL) caso algum erro tenha ocorrido.

Modos de Abertura de Arquivo

Modo	Significado
"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

Exemplo

```
// Declara o ponteiro para o descritor.
FILE *arquivo;

// Abre o arquivo exemplo.txt, que encontra-se no diretório corrente:
arquivo = fopen("exemplo.txt", "w");

// Verifica se retornou nulo (erro).
if (! arquivo) {
    printf("Erro: ... \n");
} else {
    //pode usar o arquivo aberto...
}
```

Fechando um Arquivo

•Devemos fechar os arquivos que abrimos

•Por questões de eficiência, os dados escritos são armazenados num buffer de memória e levados ao arquivo somente de tempos em tempos;

•Fechar um arquivo escreve o restante do buffer no arquivo.

•Se não o fizermos, podemos perder dados!

•A função fclose() fecha um arquivo:

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

•Exemplo:

```
fclose(arquivo); // Retorna 0 se tudo OK!
```

Escrevendo e Lendo no Arquivo

•O funcionamento de fprintf e fscanf são similares ao printf e scanf.

int fprintf(FILE *f, const char *formato, ...);

int fscanf(FILE *f, const char *formato, ...);

•Exemplo:

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

if  (arquivo) {
        fprintf(arquivo, "\nestou escrevendo...que maravilha!");
        fclose(arquivo);
} else {
        printf("\nErro ao abrir o arquivo exemplo.txt.");
}
```

Tipos de Saída Padrão

- FILE *stdin; /* Standard input stream */
- FILE *stdout; /* Standard output stream */
- FILE *stderr; /* Standard error stream */

```
fprintf(stderr, "\nAconteceu um erro!");
```

Como checar se chegou ao fim do arquivo

•Quando lemos de arquivos, é interessante saber quando o mesmo terminou (*end of file*);

•Para isso, usamos a função feof():

```
int feof(FILE *f);
```

•A função feof() retorna 0 (falso) se não é o fim do arquivo e não-zero (verdadeiro) se o fim do arquivo foi encontrado.

Lendo um arquivo linha a linha

```c
void printAllWords(FILE * fp)
{
    char tmp[20];
    int i = 1;

    while (fscanf(fp, "%19s", tmp) != EOF) {
        printf("Word %d: %s\n", i, tmp);
        i++;
    }
}
```

```c
FILE *arquivo;
char linha[1000];
int num;
arquivo = fopen("exemplo.txt", "r");
if (arquivo == NULL) {
        printf("Erro!\n");
} else {
        num = 0;
        while (!feof(arquivo)) {
                // Note: não usamos o 2o \n com scanf()!
                // Obs.: não lê linhas vazias!
                fscanf(arquivo, "%[^\n]\n", linha);
                printf("%d %s\n", num++, linha);
        }
        fclose(arquivo);
}
```

lendo_arquivo_texto.c

Lendo um arquivo linha a linha

```c
#include <stdlib.h>
#include <stdio.h>
#define FILENAME "example.txt"

int main(void) {
  /* Open the file for reading */
  char *line_buf = NULL;
  size_t line_buf_size = 0;
  int line_count = 0;
  ssize_t line_size;
  FILE *fp = fopen(FILENAME, "r");
  if (!fp)
  {
fprintf(stderr, "Error opening file '%s'\n", FILENAME);
    return EXIT_FAILURE;
  }
  /* Get the first line of the file. */
  line_size = getline(&line_buf, &line_buf_size, fp);
  /* Loop through until we are done with the file. */
  while (line_size >= 0)
  {
    /* Increment our line count */
    line_count++;
    /* Show the line details */
    printf("line[%06d]: chars=%06zd, buf size=%06zu, contents: %s",
            line_count,line_size, line_buf_size, line_buf);
    /* Get the next line */
    line_size = getline(&line_buf, &line_buf_size, fp);
  }
  /* Free the allocated line buffer */
  free(line_buf);
  line_buf = NULL;
  /* Close the file now that we are done with it */
  fclose(fp);
  return EXIT_SUCCESS;
}
```

lendo_arquivo_texto.c

Lendo um arquivo linha a linha

```c
#include <stdlib.h>
#include <stdio.h>
#define FILENAME "example.txt"

int main(void) {
  /* Open the file for reading */
  char *line_buf = NULL;
  size_t line_buf_size = 0;
  int line_count = 0;
  ssize_t line_size;
  FILE *fp = fopen(FILENAME, "r");
  if (!fp)
  {
fprintf(stderr, "Error opening file '%s'\n", FILENAME);
    return EXIT_FAILURE;
  }
  /* Get the first line of the file. */
  line_size = getline(&line_buf, &line_buf_size, fp);
  /* Loop through until we are done with the file. */
  while (line_size >= 0)
  {
    /* Increment our line count */
    line_count++;
    /* Show the line details */
    printf("line[%06d]: chars=%06zd, buf size=%06zu, contents: %s",
            line_count,line_size, line_buf_size, line_buf);
    /* Get the next line */
    line_size = getline(&line_buf, &line_buf_size, fp);
  }
  /* Free the allocated line buffer */
```

```
line[000001]: chars=000015, buf size=000016, contents: This is a file
line[000002]: chars=000010, buf size=000016, contents: which has
line[000003]: chars=000015, buf size=000016, contents: multiple lines
line[000004]: chars=000026, buf size=000032, contents: with various indentation,
line[000005]: chars=000012, buf size=000032, contents: blank lines
```

Escrita Binária: fwrite e fprintf

```
//======ESCRITA==========
FILE *pf; float pi = 3.1415, float pi2;

pf = fopen("arq.bin", "wb");
if (fwrite(&pi, sizeof(float), 1, pf) != 1)
        printf("Erro na escrita do arquivo");
fclose(pf);
```

EXEMPLO:
ESCREVE PARES E ÍMPARES EM ARQUIVOS

```c
#include<stdio.h>

int main()
{
    FILE *even, *odds;
    int n = 10;
    size_t k = 0;

    even = fopen("even.txt", "w");
    odds = fopen("odds.txt", "w");

    for(k = 1; k < n + 1; k++)
    {
        k%2==0 ? fprintf(even, "\t%5d\n", k)
               : fprintf(odds, "\t%5d\n", k);
    }
    fclose(even);
    fclose(odds);

    return 0;
}
```

Leitura Binária: fread

```
//======LEITURA==========
pf = fopen("arq.bin", "rb")
if (fread(&pi2, sizeof(float), 1, pf) != 1)
        printf("Erro na leitura do arquivo");
printf("\npi2 = %f", pi2);
fclose(pf);
```

Exemplo com structs

```
typedef struct jogador {
        int   identificador;
        char  nome[30];
        float pontos;
        int   ranking;
}Jogador;

int main() {
        FILE *arquivo;

        arquivo = fopen("exemplo.bin", "rwb");
```

Exemplo com structs

```
if (arquivo != NULL) {
        Jogador player;

        //Escrita das structs Jogador
        player.identificador = 1;
        strcpy(player.nome, "Jorge da Silva");
        player.pontos = 987.21;
        player.ranking = 12;
        fwrite(&player, sizeof(Jogador), 1, arquivo);
        //Leitura das structs Jogador
        while (!feof(arquivo)) {
                fread(&player, sizeof(Jogador), 1, arquivo);
                printf("ID: %d | NOME: %s | Pontos: %.2f | Ranking: %d\n",
                                player.identificador,
                                player.nome,
                                player.pontos,
                                player.ranking);
        }
        fclose(arquivo); // Retorna 0 se tudo OK!
} else {
        printf("\nErro ao abrir o arquivo exemplo.bin");
}
```

Movimentar-se: fseek e rewind

- precisamos de funções que mude a “posição atual” do cursor (fluxo de leitura/escrita):

  – int fseek(FILE *stream, long offset, int whence);

  /*posiciona o cursor no offset (posição), a partir do whence (ponto de partida). Retorna 0 se bem sucedido. */

  – long ftell(FILE *stream);

  /*retorna o offset da posição atual. */

  – void rewind(FILE *stream);

  /*posiciona o cursor no início do arquivo*/

Função fseek

fseek(arq, qtd, origem):

- arq = descritor do arquivo;
- qtd = número de bytes representando o tamanho do deslocamento;
- origem = de onde iremos nos delocar. Os valores aceitos encontram-se na tabela abaixo:

Nome	Valor	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Ponto corrente no arquivo
SEEK_END	2	Fim do arquivo

* Constantes definidas em `stdio.h`

Apagando um Arquivo

- Para excluir um arquivo, usamos a função remove():

```
int remove(char *nomeArquivo)
```

- A função especifica o nome do arquivo, e não um descritor (ponteiro).

Observação: Como rodar um processo

roda_processo.c

```c
#include <stdio.h>
void print_all(FILE *stream)
{
    int c;
    while ((c = getc(stream)) != EOF)
putchar(c);
}
int main(void)
{
    FILE *stream;

    if ((stream = popen("ping www.google.com -c 3", "r")) == NULL)
        return 1;
    print_all(stream);
    pclose(stream);
    return 0;
}
```

OBRIGADO!

RAFAELVC2@GMAIL.COM

https://books.goalkicker.com