**FUNÇÕES DE PROCESSAMENTO DE ARQUIVO TEXTO EM C**

A transferência de dados de um arquivo para a memória principal é feita através de um *buffer*, que é uma área da memória principal usada especialmente para essa finalidade.

Um arquivo pode ser interpretado como uma sequência de caracteres (arquivo texto) ou como uma sequência de bits (arquivo binário). Um programa fonte em C é um arquivo texto e por isso pode ser editado em um bloco de notas, por exemplo.

**Função fscanf**

**int fscanf(***input stream*, *control string*, *lista de referências para variáveis de entrada***)**

A função fscanf lê dados do arquivo de entrada, caractere por caractere e armazena nas variáveis relacionadas na lista de parâmetros. Para isso precisa receber os endereços dessas variáveis e por esse motivo os parâmetros são do tipo pointer.

O primeiro parâmetro, *input stream*, é a referência para uma estrutura denominada FILE[[1]](#footnote-1) que implementa em C um *stream* (uma sequencia de bytes).

O segundo parâmetro, *control string*, é uma cadeia de formatação usada para a conversão de cada valor lido para o respectivo tipo declarado pela variável de entrada referenciada.

Assim, se *control string* = “%d %c”, por exemplo, o primeiro valor lido será convertido para o tipo *int* e o segundo valor lido será convertido para o tipo *char* e neste caso, as referências das variáveis de entrada deverão ser, respectivamente, pointer para *int* e pointer para *char*.

Caso a formatação seja “%20s”, a variável deverá ser, por exemplo, char nome[21] que é um array de caracteres. A função fscanf, neste caso, vai ler no máximo 20 caracteres **ou** até encontrar um branco ou caractere de quebra de linha (\n).

A função fscanf, ao encontrar na entrada um espaço em branco (ou newline), pula para o próximo caractere não branco.

A função fscanf devolve um número inteiro que é igual ao número de conversões bem sucedidas ou a constante EOF com valor igual a -1, caso não exista mais nenhum caractere de entrada.

**Atenção:** para ler uma string é preciso ter um array de char como variável de entrada.

Não serve um pointer para char. Por exemplo, para fscanf(entrada,”%80s”,nome), devemos ter char nome[81], não serve char \*nome.

**Atenção:** A função fscanf C acrescenta à cadeia de caracteres que vai receber os bytes lidos no arquivo o caractere \0 que indica fim da string. No código

k = fscanf(entrada, "%80s", linha);

caso o arquivo de entrada contenha os caracteres RIOJA, a string linha vai conter:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| R | I | O | J | A | \0 |

**Função fgets**

**string fgets(***buffer*, *Maxline*, *input stream***)**

A função fgets lê uma linha do arquivo de entrada e armazena a quantidade de caracteres indicada por *Maxline* na string *buffer*. O terceiro parâmetro, *input stream*, é a referência para uma estrutura denominada FILE que implementa em C um *stream* (uma sequencia de bytes).

O parâmetro *buffer* deve ser um array de caracteres. Essa função devolve a referência desse *buffer*, isto é, um pointer.

Os caracteres existentes no arquivo de entrada são armazenados em *buffer* até que seja encontrado um caractere de quebra de linha ou que seja atingido o número máximo indicado por *Maxline* ou seja encontrado o final do arquivo.

Caso tenha chegado ao final do arquivo, devolve a constante NULL.

Por exemplo, em

p = fgets(nome,11,entrada)

no máximo 10 caracteres são lidos na entrada são copiados na string nome. O caracatere ‘\n’ (newline) também é copiado. Após o último caractere armazenado em nome, a função acrescenta o caractere ‘\0’.

Caso o arquivo de entrada contenha os caracteres RIOJA na primeira linha, a string nome vai conter:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| R | I | O | J | A | \n | \0 |

Caso a linha do arquivo de entrada não termine com newline (sem digitar a tecla <ENTER> no final da linha), a string nome vai conter os caracteres R,I,O,J,A,\0.

**EXEMPLO DE UTILIZAÇÃO DA FUNÇÃO fscanf**

#include <stdio.h> // aqui encontram-se as funções de io para arquivos e também o NULL

#include <string.h> // aqui estão as funções do tipo string

main(){

unsigned char linha[20]; // o vetor linha tem espaço para 20 caracteres - local 0 até 19

// em linha serão armazenados os caracteres que estão no arquivo

FILE \*entrada; // aqui está o pointer de stream

entrada = fopen("dados.txt","r"); // função para abrir o “arquivo” – o primeiro parâmetro

// é o nome do arquivo texto e o segundo parâmetro

// indica o modo de abertura: r indica leitura.

if (entrada == NULL) printf("\n nao encontrado \n"); // A função fopen cria a estrutura FILE

// e devolve a referência para a mesma

// ou o valor NULL

else {

printf("arquivo dados.txt encontrado - fscanf vai ler os primeiros 20 caracteres \n\n");

while(fscanf(entrada, "%20s", linha)==1){ // fscanf devolve o número de sucessos

printf(" %s ",linha); printf("\n"); // ou EOF[[2]](#footnote-2)

fclose(entrada); // fechando o arquivo que já foi lido

// A função fclose libera o espaço de FILE

}

}

**EXEMPLO DE UTILIZAÇÃO DA FUNÇÃO fgets**

#include <stdio.h> // aqui encontram-se as funções de io para arquivos e também o NULL

#include <string.h> // aqui estão as funções do tipo string

main(){

unsigned char linha[20]; // o vetor linha tem espaço para 20 caracteres - local 0 até 19

// em linha serão armazenados os caracteres que estão no arquivo

FILE \*entrada; // aqui está o pointer de stream

entrada = fopen("dados.txt","r"); // função para abrir o “arquivo” – o primeiro parâmetro

// é o nome do arquivo texto e o segundo parâmetro

// indica o modo de abertura: r indica leitura.

if (entrada == NULL) printf("\n nao encontrado \n"); // A função fopen cria a estrutura FILE

// e devolve a referência para a mesma

// ou o valor NULL

else {

printf("arquivo dados.txt encontrado fgets vai ler 10 caracteres incluindo o enter \n\n");

while(fgets(linha,11,entrada) != NULL){ // fgets guarda no array linha os 10 caracteres

// e acrescenta o \0 no final

// fgets devolve NULL se encontrar o final do arquivo

// ou a referência para linha

printf(" %s ", linha); printf("\n");

}

fclose(entrada); // fechando o arquivo que já foi lido

// A função fclose libera o espaço de FILE

}

}

1. A definição de FILE encontra-se em <stdio.h> [↑](#footnote-ref-1)
2. EOF = -1 [↑](#footnote-ref-2)