

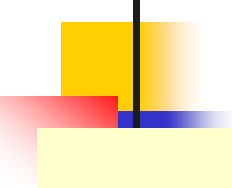
INF01202

Algoritmos e Programação

Modalidade Ensino a Distância

Linguagem C

Tópico 10: um pouco mais sobre strings - função fgets



*função **fgets**:*
lendo strings
com controle
de tamanho máximo

fgets

Lê de um arquivo uma *string* de um tamanho máximo informado.
Para leitura de *strings* a partir do dispositivo padrão de entrada,
o formato geral de *fgets* é:

fgets(string (=vetor de caracteres), tamanho máximo da string, stdin)

Além de ler *strings* de arquivos, a função *fgets* difere da função *gets* em duas questões importantes:

- 1) com *fgets*, a *string* é lida até o tamanho máximo indicado - 1 (para inserção do `\0`), com *gets* não há controle de limite da *string*, podendo ocorrer “estouro de *buffer*”;
- 2) em *fgets*, o caractere de nova linha `\n`, se dentro do tamanho máximo previsto, também integrará a *string*, em *gets* isso não ocorre.



- Leitura de uma *string* com a função
fgets

sem espaço previsto
para armazenamento
da marca de nova linha

No código a seguir, a variável (linha1) que recebe o valor digitado

via teclado foi declarada com um tamanho máximo suficiente apenas para armazenar todos os caracteres esperados, mais o \0. Não houve previsão de espaço para armazenar a marca de nova linha (\n).

O texto fornecido pelo teclado e armazenado em linha1 ficou assim igual àquele armazenado previamente em linha2. O sistema ao compará-los percebeu-os como iguais, porque continham exatamente os mesmos caracteres, na mesma ordem.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TAMLIN 13
int main ( )
{

```

```

    char linha1[TAMLIN], linha2[TAMLIN] = "arqteste.dat";
    system("color f1");
    printf("Tamanho da linha 1 (com sizeof): %d", sizeof(linha1));
    printf("\nTamanho da linha 2 (com sizeof): %d\n", sizeof(linha2));
    printf("Informe linha 1: ");
    fgets(linha1, sizeof(linha1), stdin); // caracteres lidos
    printf("Linha 1 = %s\n", linha1);
    printf("Comprimento da linha 1 (com strlen) eh %d", strlen(linha1));
    printf("\nLinha 2 = %s\n", linha2);
    printf("Comprimento da linha 2 (com strlen) eh %d \n", strlen(linha2));
    if (strcmp(linha1, linha2) == 0)
        printf("nomes iguais!!!");
    else
        printf("diferentes %s %s", linha1, linha2);
    printf("\n");
    system("pause");
    return 0;
}

```


C:\backupcida\LinguagemCPagina20082\AULA12FGETS

```

Tamanho da linha 1 (sizeof): 13
Tamanho da linha 2 (sizeof): 13
Informe linha 1: arqteste.dat
Linha 1 = arqteste.dat
Comprimento da linha 1 (strlen) eh 12
Linha 2 = arqteste.dat
Comprimento da linha 2 (strlen) eh 12
nomes iguais!!!
Pressione qualquer tecla para continuar. . .

```

Texto lido com *fgets* armazenado sem marca de nova linha.



Leitura de uma *string* com *fgets*
com espaço previsto
para armazenamento
da marca de nova linha

No código a seguir, a variável (linha1) que recebe o texto via teclado foi declarada com um tamanho que permitiu o armazenamento da marca de nova linha.

O texto fornecido pelo teclado e armazenado em linha1 ficou assim com um caractere a mais do que o armazenado previamente em linha2. O sistema ao compará-los percebeu-os então como diferentes.


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TAMLIN 14
int main ( )
```

```
{
    char linha1[TAMLIN], linha2[TAMLIN] = "arqteste.dat";
    system("color f1");
    printf("Tamanho da linha 1 (com sizeof): %d", sizeof(linha1));
    printf("\nTamanho da linha 2 (com sizeof): %d\n", sizeof(linha2));
    printf("Informe linha 1: ");
    fgets(linha1, sizeof(linha1), stdin); // caracteres lidos
    printf("Linha 1 = %s", linha1);
    printf("Comprimento da linha 1 (com strlen) eh %d", strlen(linha1));
    printf("\nLinha 2 = %s\n", linha2);
    printf("Comprimento da linha 2 (com strlen) eh %d \n", strlen(linha2));
    if (strcmp(linha1, linha2) == 0)
        printf("nomes iguais!!!");
    else
        printf("diferentes %s %s", linha1, linha2);
    printf("\n");
    system("pause");
    return 0;
}
```

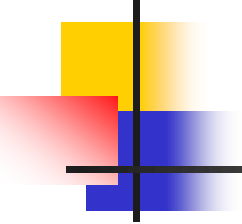
```
C:\backupcida\LinguagemCPagina20082\AULA12FGETS
Tamanho da linha 1 (sizeof): 14
Tamanho da linha 2 (sizeof): 14
Informe linha 1: arqteste.dat
Linha 1 = arqteste.dat
Comprimento da linha 1 (strlen) eh 13
Linha 2 = arqteste.dat
Comprimento da linha 2 (strlen) eh 12
diferentes arqteste.dat
arqteste.dat
Pressione qualquer tecla para continuar. . .
```

Texto lido com *fgets* armazenado com marca de nova linha.



Eliminação

da marca de nova linha
de uma *string* lida com *fgets*



No código a seguir, eliminou-se o caractere de nova linha da *string* lida com *fgets*.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TAMLIN 14
int main ( )
{

```

```

    char linha1[TAMLIN], linha2[TAMLIN] = "arq";
    system("color f1");
    printf("Tamanho da linha 1 (com sizeof): %d", sizeof(linha1));
    printf("\nTamanho da linha 2 (com sizeof): %d\n", sizeof(linha2));
    printf("Informe linha 1: ");
    fgets(linha1, sizeof(linha1), stdin); // caracteres lidos
    printf("Linha 1 = %s", linha1);
    printf("Comprimento da linha 1 (com strlen) eh %d", strlen(linha1));
    if (linha1[strlen(linha1) - 1] == '\n')
        linha1[strlen(linha1) - 1] = '\0';
    printf("\nComprimento da linha 1 (com strlen) eh %d", strlen(linha1));
    printf("\nLinha 2 = %s\n", linha2);
    printf("Comprimento da linha 2 (com strlen) eh %d \n", strlen(linha2));
    if (strcmp(linha1, linha2) == 0)
        printf("nomes iguais!!!");
    else
        printf("diferentes %s %s", linha1, linha2);
    printf("\n");
    system("pause");
    return 0;
}

```

```

C:\backupcida\LinguagemCPagina20082\AULA12\FGETSCidavt
Tamanho da linha 1 (com sizeof): 14
Tamanho da linha 2 (com sizeof): 14
Informe linha 1: arq
Linha 1 = arq
Comprimento da linha 1 (com strlen) eh 4
Comprimento da linha 1 (com strlen) e sem \n eh 3
Linha 2 = arq
Comprimento da linha 2 (com strlen) eh 3
nomes iguais!!!
Pressione qualquer tecla para continuar. . . _

```

Eliminando o \n de uma *string* lido com *fgets*

Relembrando:

strlen devolve o número de caracteres de uma *string* (vetor de caracteres), sem considerar o `\0`.

Os índices para acessar um vetor iniciam em 0.

Assim a tentativa de usar como índice, direto, o valor de retorno da função *strlen* de uma *string* para acessar a mesma *string* gera uma situação de índice inválido.

Exemplo de uso ok de *strlen*:

```
linha1[strlen(linha1) - 1] = '\0';
```