Lendo uma Linha em C

Prof. Dr. Anderson Bessa da Costa
Algoritmos e Programação 2
FACOM, Universidade Federal de Mato Grosso do Sul

1 Introdução

Ao iniciarmos na programação na linguagem de programação C, uma das primeiras dificuldades consiste em realizar a leitura de uma linha inteira. Embora nos seja ensinado o uso do código de formatação `%s` na função `scanf()` para ler strings, esse operador faz a leitura de caracteres até que o primeiro espaço em branco ou final de linha seja encontrado.

Algumas alternativas existem para ler uma linha inteira em C. Neste material trabalharemos as alternativas existentes, abordando as suas diferenças, vantagens e desvantagens.

2 Motivação

Considere o código abaixo:

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5    char str[25];
6
7    printf("Entre com o nome de seu estado: ");
8    scanf("%s", str);
9
10   printf("O estado digitado eh %s.", str);
11   printf(" O tamanho da string eh lida eh %lu.", strlen(str));
12
13   return 0;
14 }
```

Considere a seguinte saída:

```
Entre com o nome de seu estado: Mato Grosso do Sul
O estado digitado eh Mato. O tamanho da string eh lida eh 4.
```

O código apresentado solicita ao usuário a entrada do nome de um estado. O problema é que ao entrar com um nome que seja composto (separado por espaço), utilizando o código de formatação `%s` na função `scanf()` somente o primeiro nome é armazenado. Isto pode ser observado na impressão na saída e também no comprimento da string como 4, obtido com a função `strlen()`.

3 Alternativa 1: gets

O primeiro *worm* da internet (o *Morris Internet Worm*) escapou há mais de 30 anos, mais precisamente em 2 de novembro de 1988. O *worm* em questão utilizava `gets()` e um **buffer overflow** como um de seus métodos para propagar de sistema para sistema. O problema básico é que a função `gets()` não sabe quão grande o buffer é, e continua a leitura até que seja encontrado uma nova linha ou EOF, e pode transbordar os limites do buffer fornecido.

Você deve esquecer que `gets()` existiu. O padrão C11 ISO/IEC 9899:2011 eliminou o `gets()` como função padrão (na verdade foi marcada como 'obsoleta' e 'decreptada' no ISO/IEC 9899:1999/Cor.3:2007 - *Technical Corrigendum 3 for C99*, e então removida em C11). Infelizmente, essa função será mantida em bibliotecas por muitos anos (na verdade décadas) por questões de retrocompatibilidade.

4 Alternativa 2: fgets (RECOMENDADO)

Uma linha pode ser lida com a função `fgets` presente na biblioteca `stdio`. A sua assinatura é:

```
char *fgets (char *str, int num, FILE *stream)
```

Obter string do stream Lê caracteres do stream e os armazena como uma string C em *str* até que $num - 1$ caracteres tenham sido lidos ou uma nova linha ou o final do arquivo seja alcançado, o que acontecer primeiro.

Um caracter nova linha faz `fgets` parar a leitura, mas é considerado um caractere válido pela função e incluído na string copiada para *str*.

O caractere NULL (\0) é automaticamente adicionado ao final dos caracteres copiados para *str*.

Note que `fgets` é bem diferente de `gets`: não apenas `fgets` aceita um stream como argumento, mas também permite especificar o tamanho máximo de *str* e inclui na string qualquer caracter final de nova linha.

4.1 Exemplo

Considere o código abaixo:

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5    char str[25];
6
7    printf("Entre com o nome de seu estado: ");
8    fgets(str, 25, stdin);
9
10   printf("O estado digitado eh %s.", str);
11   printf(" O tamanho da string eh lida eh %lu.", strlen(str));
12   return 0;
13 }
```

A saída do programa:

```
Entre com o nome de seu estado: Mato Grosso do Sul
O estado digitado eh Mato Grosso do Sul
. O tamanho da string eh lida eh 19.
```

Algumas observações:

- Para lermos do teclado precisamos usar o stream `stdin`;
- Como pode ser observado, foi armazenado o caractere de nova linha na string. A string possui 18 letras (Mato Grosso do Sul) + 1 caractere de nova linha, totalizando assim 19. Além disso, é utilizado mais 1 posição adicional para armazenar o NULL (\0), mas esse caractere não é contabilizado por `strlen()`;
- Um vetor de tamanho n pode armazenar no máximo $n-1$ caracteres lidos. Isso devido que sempre um caractere é utilizado para guardar o NULL (\0) para indicar fim da string;

Existe uma solução simples para remover um possível caractere de nova linha lido. Basta adicionar a seguinte linha de código após o uso do `fgets`:

```
str[strcspn(str, "\n")] = '\0';
```

Explicação A função `strcspn()` calcula o comprimento do número de caracteres antes da primeira ocorrência do caractere presente em ambas strings. Uma vez que passamos "$\backslash n$"como segunda string nós iremos obter o comprimento da string antes da ocorrência de "$\backslash n$. Nós iremos agora colocar o '\0' no lugar do '$\backslash n$'.

4.2 Exemplo Modificado: Removendo caractere nova linha

Considere o código abaixo:

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5    char str[25];
6
7    printf("Entre com o nome de seu estado: ");
8    fgets(str, 25, stdin);
9    str[strcspn(str, "\n")] = '\0';
10
11   printf("O estado digitado eh %s.", str);
12   printf("O tamanho da string eh lida eh %lu.", strlen(str));
13
14   return 0;
15 }
```

A saída do programa:

```
Entre com o nome de seu estado: Mato Grosso do Sul
O estado digitado eh Mato Grosso do Sul. O tamanho da string eh lida eh 18.
```

Com isso conseguimos ler uma linha inteira e removendo um possível caractere de nova linha armazenado também. Concluímos com sucesso nosso objetivo.

5 Alternativa 3: scanf adaptado

É possível conseguirmos fazer a leitura de uma linha inteira utilizando a própria função `scanf()`. Para isso, devemos utilizar um comando que se assemelha a uma expressão regular:

```
scanf("%[^\n]s", str);
```

Essa linha pode ser lida como "ler uma string e armazená-la em str até que seja digitado enter (ou seja, encontrado uma nova linha)".

5.1 Exemplo

Considere o código abaixo:

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5    char str[25];
6
7    printf("Entre com o nome de seu estado: ");
8    scanf("%[^\n]s", str);
9
10   printf("O estado digitado eh %s.", str);
11   printf(" O tamanho da string eh lida eh %lu.", strlen(str));
12   return 0;
13 }
```

A saída do programa:

```
Entre com o nome de seu estado: Mato Grosso do Sul
O estado digitado eh Mato Grosso do Sul. O tamanho da string eh lida eh 18.
```

Como observado, a string é lida até o final da linha e armazenada em *str*. Ao encontrar uma nova linha, a leitura para e não é armazenado o caractere de nova linha em *str*.

Entretanto em termos de segurança não existe diferença entre o `scanf()` e o `gets()`, uma vez que ambos lêem da entrada padrão e podem muito bem transbordar a mensagem, se o usuário entrar com mais dados que o buffer fornecido suporta.