Safe String Manipulation

June 17, 2017

Dană Andrei - Iulian

Computers and Information Technology
(English Language)

Group: CEN 11B

1st Year

Introduction

Strings are one-dimensional array of characters terminated by a null character. Thus a null-terminated string contains the characters that comprise the string followed by a null.

String manipulation functions have been created in order to allow users to work with strings in an easier manner, such as finding their length, lexicographically compare them, copy them and so on.

Problem Statement

Create a library for the most common safe string manipulation functions. The library covers the following functions:

- **strlen** - Returns the length of a string
- **strcpy** - Copies a string into another string
- **strncpy** - Copies a number of characters from a string into another one
- **strcat** - Pastes a string at the end of another string
- **strcmp** - Lexicographically compares two strings
- **gets** - Reads characters from keyboard, including spaces
- **strchr** - Returns the first apparition of a character in a string
- **strstr** - Returns the first apparition of a character in another string

Pseudocode

The following section will present the pseudocode of the functions used for the project. The string manipulation functions are included in the `string_manipulation` C file. The random input generator function is included in the `input_generator` C file.
Here are the the most important functions used:

1. `my_strlen`

```
my_strlen (char givenString[])
1.      int letterCounter
2.      letterCounter <- 0
3.      while givenString[letterCounter] != '\0'
4.              letterCounter <- letterCounter + 1
5.      return letterCounter
```

2. `my_strcpy`

```
my_strcpy (char targetString[], char givenString[])
1.      int iterator
2.      iterator <- 0
3.      while 1
4.              targetString[iterator] <- givenString[iterator]
5.              if givenString[iterator] == '\0'
6.                      break
7.              iterator <- iterator + 1
8.      int letterCounter
9.      lettercounter <- 0
10.     while givenString[letterCounter] != '\0'
11.             letterCounter <- letterCounter + 1
12.     targetString[letterCounter] <- '\0'
```

3. my_gets

```
my_gets (char givenString[])
        char inputCharacter
        int iterator
        iterator <- 0
        while inputCharacter != EOF
                inputCharacter <- getc(stdin)
                if inputCharacter == '\n'
                        givenString[iterator] <- '\0'
                        return
                givenString[iterator] <- inputCharacter
                iterator <- iterator + 1
        givenString[iterator - 1] <- '\0'
        iterator <- 0;
```

4. my_strchr

```
*my_strchr (char givenString[], char inputCharacter)
        int iterator
        iterator <- 0
        while givenString[iterator] != '\0'
                if givenString[iterator] == inputCharacter
                        return givenString + iterator
                iterator <- iterator + 1
```

5. random_string_generator

```
*random_string_generator(char *str, size_t size)
        const char charset[]
        charset<-"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
        if size
                size <- size - 1
                size_t n
                n <- 0
                for n <- 0 to size - 1 do
                        int key
                        key <- rand() % (int) (sizeof charset - 1)
                        str[n] <- charset[key]
                str[size] <- '\0'
        return str
```

Application design

The library for the string manipulation functions contains the header **string_manipulation.h** which has all the prototypes for the functions used in the project to manipulate strings:

int my_strlen (char givenString[]);
void my_strcpy (char targetString[], char givenString[]);
void my_strncpy(char targetString[],char givenString[],int numberOfCharacters);
void my_strcat (char targetString[], char givenString[]);
int my_strcmp (char firstString[], char secondString[]);
void my_gets (char givenString[]);
char *my_strchr (char givenString[], char inputCharacter);
char *my_strstr (char targetString[], char givenString[]);

The library for random input generating contains the header **input_generator.h** which has the prototype for the function that will be used to generate random strings:

char *random_string_generator(char *str, size_t size)

Functions summary:

1. my_strlen

This function returns the number of characters a given string has (the null character is not counted). It uses a counter variable that is incremented as every character in the string is visited. After the process is ended, the counter is returned.

2. my_strcpy

This function will copy all the characters from one string into another. It copies character by character from the given string into the target string, the process being stopped when the all the characters from the string we want to copy from have been copied. After the copying process, the function will search for the position of the last copied character and ends the string by placing the null character at the end of the string.

3. my_strncpy

This function will copy a certain number of characters from one string into another. It works almost exactly like my_strcpy, only this time the copying process is stopped when either the given number of characters has been copied or when the string we copy from ends.

4. my_strcat

This function will paste a string at the end of another string. It first searches for the end of the target string and then it starts the copying process. It stops when all the characters have been successfully copied. Before the function exits, it places the null character at the end of the target string.

5. my_strcmp

This function will lexicographically compare two given string and return 1, 0 or -1, depending on the result. It compares character by character the two strings and if at a certain step they become different, the function will return 1 if the first string is greater than the second one, or -1 if it is smaller. If the two strings have been browsed and no value was returned, the function will return 0.

6. my_gets

This function will read and memorize in a string characters (spaces included) until the user presses the ENTER key. It reads character by character, memorizing that character in a char variable, then copying it in the string. When the ENTER character is met, the process stops.

7. my_strchr

This function will return the first apparition of a character in a given string. It browses the string character by character and when that specific character is found, the process is stopped and the adress is returned. If the character can't be found inside the string, the function will return a null value.

8. my_strstr

This function will return the first apparition of a string in another string. The function will start to search the first character of the searched string in the target string. If it finds it, it checks if the rest of the characters from the searched string are in the correct order. If they are, the adress is returned, if not, the process continues. If the function does not find the given string it will return a null value.

Source Code

```
//----------------------------string_manipulation.h-------------------------------
int my_strlen (char givenString[]);

void my_strcpy (char targetString[], char givenString[]);

void my_strncpy (char targetString[], char givenString[], int
    numberOfCharacters);

void my_strcat (char targetString[], char givenString[]);

int my_strcmp (char firstString[], char secondString[]);

void my_gets (char givenString[]);

char *my_strchr (char givenString[], char inputCharacter);

char *my_strstr (char targetString[], char givenString[]);
```

```
//----------------------------string_manipulation.c-------------------------------
# include <stdio.h>
# include <stdlib.h>
# include "string_manipulation.h"


int my_strlen (char givenString[]) {

       int letterCounter = 0;

       while (givenString[letterCounter] != '\0') {

              letterCounter++;

       }

       return letterCounter;

}

void my_strcpy (char targetString[], char givenString[]) {

       int iterator = 0;

       while (1) {

              targetString[iterator] = givenString[iterator];
```

```
                if (givenString[iterator] == '\0') {

                        break;

                }

                iterator++;

        }

        int letterCounter = 0;

        while (givenString[letterCounter] != '\0') {

                letterCounter++;

        }

        targetString[letterCounter] = '\0';

}


void my_strncpy (char targetString[], char givenString[], int
    numberOfCharacters) {

        int iterator = 0;

        if (numberOfCharacters == 0) {

                return;

        }

        while (iterator < numberOfCharacters) {

                targetString[iterator] = givenString[iterator];

                if (givenString[iterator] == '\0') {

                        break;

                }

                iterator++;

        }

        targetString[iterator] = '\0';
```

```
}


void my_strcat (char targetString[], char givenString[]) {

        int iterator = 0;;
        int givenStringLegth = 0;
        int targetStringLegth = 0;

        while (targetString[iterator++] != '\0') {

                targetStringLegth++;

        }

        for (iterator = 0; givenString[iterator] != '\0'; iterator++) {

                targetString[targetStringLegth + iterator] =
                    givenString[iterator];

        }

        if (targetString[targetStringLegth + iterator] != '\0') {

                targetString[targetStringLegth + iterator] = '\0';

        }

}

int my_strcmp (char firstString[], char secondString[]) {

        int iterator = 0;

        while (1) {

                if ( (int)(firstString[iterator]) >
                    (int)(secondString[iterator]) ) {

                        return 1;

                }
                else {

                        if ( (int)(firstString[iterator]) <
                            (int)(secondString[iterator]) ) {

                                return -1;

                        }
```

```
                }

                if (firstString[iterator] == '\0' ||
                    secondString[iterator] == '\0') {

                        break;

                }

                iterator++;

        }

        return 0;

}

void my_gets (char givenString[]) {

        char inputCharacter;
        int iterator = 0;

        while (inputCharacter != EOF) {

                inputCharacter = getc(stdin);

                if (inputCharacter == '\n') {

                        givenString[iterator] = '\0';

                        return;
                }

                givenString[iterator] = inputCharacter;

                iterator++;

        }

        givenString[iterator - 1] = '\0';

        iterator = 0;

}


char *my_strchr (char givenString[], char inputCharacter) {

        int iterator = 0;
```

```
        while (givenString[iterator] != '\0') {

                if (givenString[iterator] == inputCharacter) {

                        return givenString + iterator;

                }

                iterator++;

        }

}


char *my_strstr (char targetString[], char givenString[]) {

        int targetStringIterator = 0;
        int givenStringIterator = 0;
        int position = my_strlen(targetString);

        if (my_strlen(targetString) < my_strlen(givenString)) {

                return targetString + my_strlen(targetString);

        }

        while (targetString[targetStringIterator] != '\0') {

                if (targetString[targetStringIterator] == givenString[0])
                    {

                        position = targetStringIterator;

                        while (givenString[givenStringIterator] != '\0') {

                                if (targetString[targetStringIterator] !=
                                    givenString[givenStringIterator]) {

                                        givenStringIterator = 0;
                                        targetStringIterator = position;
                                        position = my_strlen(targetString);
                                        break;

                                }

                                givenStringIterator++;
                                targetStringIterator++;
```

```
                }

            }

            targetStringIterator++;

        }

        return targetString + position;

}
```

```
//-----------------------------input_generator.h-------------------------------
char *random_string_generator(char *str, size_t size);
```

```
//-----------------------------input_generator.c-------------------------------
# include <stdio.h>
# include <stdlib.h>
# include <time.h>
# include "input_generator.h"

char *random_string_generator(char *str, size_t size) {

        const char charset[] =
            "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ";

            if (size) {

                        --size;

                        size_t n = 0;

                        for (n = 0; n < size; n++) {

                                int key = rand() % (int) (sizeof
                                    charset - 1);
                                str[n] = charset[key];

                        }

                        str[size] = '\0';

            }

        return str;
}
```

```c
//----------------------------main.c------------------------------
# include <stdio.h>
# include <stdlib.h>
# include <stdbool.h>
# include "string_manipulation.h"
# include "input_generator.h"

# define DIM 10010

char first_string[DIM];
char second_string[DIM];
char temp_string[DIM];
char temp_character;

int random_string_length;
int task_type;
int task_option;

int main()
{

        printf("Enter the length for the first string: ");
        scanf("%d", &random_string_length);
        my_strcpy(first_string, random_string_generator(temp_string,
            random_string_length));

        printf("Enter the length for the second string: ");
        scanf("%d", &random_string_length);
        my_strcpy(second_string, random_string_generator(temp_string,
            random_string_length));
        system("cls");
        printf("\nString 1: %s\n", first_string);
        printf("\nString 2: %s\n", second_string);
        printf("\n================================================================================\n\n");
        printf("*** Task menu ***\n");
        printf("\t1. Print legth\n");
        printf("\t2. Copy\n");
        printf("\t3. Copy number of characters\n");
        printf("\t4. Concatenate\n");
        printf("\t5. Lexicographically compare\n");
        printf("\t6. Get the first apparition of a character\n");
        printf("\n================================================================================\n");
        printf("\nEnter the task number: ");
        scanf("%d", &task_type);

        system("cls");

        switch (task_type) {
```

```
case 1: { /* Print length instructions START */

        printf("> Task options:\n\n");
        printf("\t1. Length of the first string\n");
        printf("\t2. Length of the second string\n");
        printf("\t3. Length of both strings\n");
        printf("\nEnter the option number: ");
        scanf("%d", &task_option);

        if (task_option == 1) {

                printf("\nLength of the first string is: %d
                    + 1\n", my_strlen(first_string));

        }
        else {

                if (task_option == 2) {

                        printf("\nLength of the second
                            string is: %d + 1\n",
                            my_strlen(second_string));

                }
                else {

                        if (task_option == 3) {

                            printf("\nLength of the first
                                string is: %d + 1\n",
                                my_strlen(first_string));
                            printf("\nLength of the second
                                string is: %d + 1\n",
                                my_strlen(second_string));

                        }
                        else {

                                printf("\n\nNo such option.
                                    Exiting...\n");
                                return 0;

                        }

                }

        }

}
break; /* Print length instructions END */
```

```
case 2: { /* Copy instructions START */

        printf("> Task options:\n\n");
        printf("\t1. Copy the first string into the second
            string\n");
        printf("\t2. Copy the second string into the first
            string\n");
        printf("\nEnter the option number: ");
        scanf("%d", &task_option);

        if (task_option == 1) {

                system("cls");

                printf("Results:\n\n");
                printf("> Before:\n\n");
                printf("\tFirst String: %s\n",
                    first_string);
                printf("\tSecond String: %s\n",
                    second_string);

                my_strcpy(second_string, first_string);

                printf("\n> After:\n\n");
                printf("\tFirst string: %s\n",
                    first_string);
                printf("\tSecond string: %s\n",
                    second_string);

        }
        else {

                if (task_option == 2) {

                        system("cls");

                        printf("Results:\n\n");
                        printf("> Before:\n\n");
                        printf("\tFirst String: %s\n",
                            first_string);
                        printf("\tSecond String: %s\n",
                            second_string);

                        my_strcpy(first_string,
                            second_string);

                        printf("\n> After:\n\n");
                        printf("\tFirst string: %s\n",
                            first_string);
```

```
                        printf("\tSecond string: %s\n",
                            second_string);

                }
                else {

                        printf("\n\nNo such option.
                            Exiting...\n");
                        return 0;

                }

        }

}
break; /* Copy instructions END */

case 3: { /* Copy number of characters instructions START
    */

        system("cls");

        int temp_number_of_characters;

        printf("Enter the number of characters to copy: ");
        scanf("%d", &temp_number_of_characters);

        system("cls");

        printf("> Task options:\n\n");
        printf("Copy the first %d characters from:\n",
            temp_number_of_characters);
        printf("\t1. The first string into the second
            string\n");
        printf("\t2. The second string into the first
            string\n");
        printf("\nEnter the option number: ");
        scanf("%d", &task_option);

        if (task_option == 1) {

                system("cls");

                printf("Results:\n\n");
                printf("> Before:\n\n");
                printf("\tFirst String: %s\n",
                    first_string);
                printf("\tSecond String: %s\n",
                    second_string);
```

```
                my_strncpy(second_string, first_string,
                    temp_number_of_characters);

                printf("\n> After:\n\n");
                printf("\tFirst string: %s\n",
                    first_string);
                printf("\tSecond string: %s\n",
                    second_string);

        }
        else {

                if (task_option == 2) {

                        system("cls");

                        printf("Results:\n\n");
                        printf("> Before:\n\n");
                        printf("\tFirst String: %s\n",
                            first_string);
                        printf("\tSecond String: %s\n",
                            second_string);

                        my_strncpy(first_string,
                            second_string,
                            temp_number_of_characters);

                        printf("\n> After:\n\n");
                        printf("\tFirst string: %s\n",
                            first_string);
                        printf("\tSecond string: %s\n",
                            second_string);

                }
                else {

                        printf("\n\nNo such option.
                            Exiting...\n");
                        return 0;

                }

        }

}
break; /* Copy number of characters instructions END */

case 4: { /* Concatenate instructions START */

        system("cls");
```

```
printf("> Task options:\n\n");
printf("\t1. Concatenate the first string at the
    end of the second string\n");
printf("\t1. Concatenate the second string at the
    end of the first string\n");
printf("\nEnter the option number: ");
scanf("%d", &task_option);

if (task_option == 1) {

        system("cls");

        printf("Results:\n\n");
        printf("> Before:\n\n");
        printf("\tFirst String: %s\n",
            first_string);
        printf("\tSecond String: %s\n",
            second_string);

        my_strcat(second_string, first_string);

        printf("\n> After:\n\n");
        printf("\tFirst string: %s\n",
            first_string);
        printf("\tSecond string: %s\n",
            second_string);

}
else {

        if (task_option == 2) {

                system("cls");

                printf("Results:\n\n");
                printf("> Before:\n\n");
                printf("\tFirst String: %s\n",
                    first_string);
                printf("\tSecond String: %s\n",
                    second_string);

                my_strcat(first_string,
                    second_string);

                printf("\n> After:\n\n");
                printf("\tFirst string: %s\n",
                    first_string);
                printf("\tSecond string: %s\n",
                    second_string);
```

```
                }
                else {

                        printf("\n\nNo such option.
                            Exiting...\n");
                        return 0;

                }

        }

}
break; /* Concatenate instructions END */

case 5: { /* Lexicographically compare instructions START
    */

        printf("String 1: %s\n\n", first_string);
        printf("String 2: %s\n\n", second_string);

        printf("Comparison result: %d",
            my_strcmp(first_string, second_string));

}
break; /* Lexicographically compare instructions END */

case 6: { /* First apparition of a character instructions
    START */

        system("cls");

        printf("> Task options:\n\n");
        printf("Select the string to work with:\n\n");
        printf("\t1. First string\n");
        printf("\t2. Second string\n");
        printf("\nEnter the option number: ");
        scanf("%d", &task_option);

        system("cls");
        if (task_option == 1) {

                printf("First string: %s\n\n",
                    first_string);

        }
        else {

                printf("Second string: %s\n\n",
                    second_string);
```

```
                }

                printf("Enter the character to search: ");

                scanf(" %c", &temp_character);

                system("cls");

                printf("Results:\n\n");
                if (task_option == 1) {

                        printf("First string: %s\n\n",
                            first_string);

                }
                else {

                        printf("Second string: %s\n\n",
                            second_string);

                }

                if (task_option == 1) {

                        printf("First apparitions of character:
                            %s\n\n", my_strchr(first_string,
                            temp_character));

                }
                else {

                        printf("First apparitions of character:
                            %s\n\n", my_strchr(second_string,
                            temp_character));

                }

        }
        break; /* First apparition of a character instructions
            END */

        default: printf("\nError. No such option ...\n");

    }

    return 0;
}
```

Experiments and results

```
string_1 = "sadsunhusg ergeur ge e eug eg e geuyg"
string_2 = " asfhjd yrwewswredgp p p gdgdg"
string_3 = "hg usiou sgqwfrowe'dfi w fkirgfwgferigrig"
string_4 = "edfrghujikd oisdrfgp i ghiregh g"
string_5 = " wrsqacdsdejw hfdoshngv rgfuio ruegfreugieg"
string_6 = "rfdghjoegfrijg gurger euetr ggh u hthth"
string_7 = " ghhjwasr ae"
string_8 = "avbdsaj fbf fhbdhfdh"
string_9 = "wqfrefg girugguidsfioaawr r r gfrgdg"
string_10 = "jlasfhas ff kgadsfkdskfgte"


my_strlen(string_5): 43 + 1
my_strlen(string_2): 32 + 1
my_strlen(string_7): 12 + 1


my_strcpy(string_3, string_1)
    string_3: "sadsunhusg ergeur ge e eug eg e geuyg"
my_strcpy(string_10, string_8)
    string_10: "avbdsaj fbf fhbdhfdh"
my_strcpy(string_5, string_2)
    string_5: " asfhjd yrwewswredgp p p gdgdg"


my_strncpy(string_4, string_1, 15)
    string_4: "sadsunhusg erge"
my_strncpy(string_9, string_3, 54)
    string_9: "hg usiou sgqwfrowe'dfi w fkirgfwgferigrig"
my_strncpy(string_6, string_10, 0)
    string_6: "rfdghjoegfrijg gurger euetr ggh u hthth"


my_strcat(string_1, string_6)
    string_1: "sadsunhusg ergeur ge e eug eg e geuygrfdghjoegfrijg
        gurger euetr ggh u hthth"
my_strcat(string_2, string_8)
    string_2: " asfhjd yrwewswredgp p p gdgdgavbdsaj fbf fhbdhfdh"
my_strcat(string_9, string_3)
    string_9: "wqfrefg girugguidsfioaawr r r gfrgdghg usiou
        sgqwfrowe'dfi w fkirgfwgferigrig"


my_strcmp(string_4, string_6)
    Result: -1
my_strcmp(string_1, string_8)
    Result: 1
```

```
my_strcmp(string_9, string_3)
    Result: 1


my_strchr(string_2, 'd')
    Result: "d yrwewswredgp p p gdgdg"
my_strchr(string_7, 'w')
    Result: "wasr ae"
my_strchr(string_8, 'x')
    Result: <null>


my_strstr(string_1, "crweyufg rg")
    Result: <null>
my_strstr(string_3, "g usi")
    Result: "g usiou sgqwfrowe'dfi w fkirgfwgferigrig"
my_strstr(string_10, string_1)
    Result: <null>
```

Conclusions

Working on this project, I can say that I now have a better understanding of the basic string manipulation functions. These types of functions have a good use all over the internet: data bases, online forms and even games at a certain extent. I can say that it was a good experience creating these functions from scratch as I gained a lot of knowledge about them and about C language in general: how to work with makefiles, cmd, functions, etc.

References

1. www.stackoverflow.com

2. www.cplusplus.com

3. www.sharelatex.com

GitHub Link

https://github.com/Evohunt/Safe-String-Manipulation