Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра вычислительных систем

ОТЧЕТ
по практической работе 4

по дисциплине «**Программирование**»

Выполнил:
студент гр. ИВ-222
«17» мая 2023 г. _____ /Городилов.П.А./

Проверил:
Старший преподаватель Кафедры ВС
«18» мая 2023 г. _____ /Фульман.В.О./

Оценка «_____»

Новосибирск 2023

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

В рамках лабораторной работы необходимо реализовать обработку строковой информации в соответствии с заданием и реализовать необходимые функции. В качестве обрабатываемых данных рассматриваются пути к файлам в операционных системах Windows и GNU/Linux. Вариант 6: преобразовать все Windows-пути формата Cyrgwin к оригинальным Windows-путям.

ВЫПОЛНЕНИЕ РАБОТЫ

Сперва было необходимо реализовать функции стандартного строкового заголовочного файла string.h.(string.c strings.h)

Функция slen принимает на вход строку, считает длину этой строки и возвращает её.

```
size_t slen(char *str)//+
{
    size_t len = 0;
    for (int i = 0; str[i] != '\0'; i++)
        len++;
    return len;
}
```

Функция schr принимает на вход строку и символ, возвращает указатель на символ в строке, если он есть, если его нет – NULL.

```
char *schr(char *str, char c)//+
{
    while (*str != c && *str != '\0')
        str++;
    if (*str == c)
        return str;
    else
        return NULL;
}
```

Функция scmp принимает на вход 2 строки, возвращает 0 если сравниваемые строки идентичны, положительное число если строки отличаются и ASCII код первого отличающегося символа в первой строке больше кода символа на той же позиции во второй строке, отрицательное число если строки отличаются и ASCII код первого отличающегося символа в первой строке меньше кода символа на той же позиции во второй строке.

```
int scmp(char *str1, char *str2)//+
{
    while (*str1)
    {
        if (*str1 != *str2)
            break;
        str1++;
        str2++;
    }
    if((*str1 - *str2)>0)
        return 1;
    else if((*str1 - *str2)<0)
        return -1;
    return 0;
}
```

Функция scpy принимает на вход 2 строки, копирует содержание второй строки в первую, возвращает строку, в которую были скопированы символы.

```
char *scpy(char *str1, char *str2)//+
{
    if (str1 == NULL)
    {
        return NULL;
    }
    int len = slen(str1);
    for(int i = 0; i < len; i++){
        str2[i] = str1[i];
    }
    str2[len] = '\0';
    return str2;
}
```

Функция is_del принимает на вход символ и строку, возвращает 1, если символ входит в строку, иначе – 0.

```
int is_del(char c, char *delim)
{
    while (*delim != '\0')
    {
        if (c == *delim)
            return 1;
        delim++;
    }
    return 0;
}
```

Функция stok принимает на вход 2 строки, возвращает первую подстроку до разделителя, если разделитель не встречен - возвращает NULL.

```
char* stok(char *str, char *delim)//+
{
    static char *p;
    if (!str)
    {
        str = p;
    }
    if (!str)
    {
        return NULL;
    }
    while (1)
    {
        if (is_del(*str, delim))
        {
            str++;
            continue;
        }
        if (*str == '\0')
        {
            return NULL;
        }
        break;
    }
    char *ret = str;
    while (1)
    {
        if (*str == '\0')
        {
            p = str;
            return ret;
        }
        if (is_del(*str, delim))
        {
            *str = '\0';
            p = str + 1;
            return ret;
        }
        str++;
    }
}
```

Функция scat принимает на вход 2 строки, дописывает в конец первой строки вторую строку и возвращает указатель на получившуюся строку.

```
char *scat(char *str1, char *str2)
{
    char *ptr = str1 + slen(str1);
    while (*str2 != '\0')
    {
        *ptr++ = *str2++;
    }
    *ptr = '\0';
    return str1;
}
```

Функция с delchar принимает на вход строку и удаляет из неё первый элемент.

```
void delchar(char *str){
    for(int i=0;i<slen(str);++i)
        str[i]=str[i+1];
}
```

Функция s_upper принимает на вход строку и переводит буквы в верхний регистр.

```
char* s_upper(char* str){
    str[0]=str[0]-32;
    return str;
}
```

Также необходимо было реализовать функции, обеспечивающие работу приложения. (lib.c lib.h)

Функция input позволяет считать и записать данные в строку, передаваемую в аргументе.

```
char *input(char *argument)
{
    scanf("%s", argument);
    return argument;
}
```

Функция symbols_check проверяет, чтобы все символы в передаваемой строке были корректные.

```
int symbols_check(char *argument)
{
    for (char *a = argument; *a != '\0'; a++)
    {
        if ((*a == '\\')||(*a == ':')||(*a == '*')||(*a == '?')||(*a == '<')||(*a == '>')||(*a == '|')||(*a == '«'))
            return -1;
    }
    return 0;
}
```

Функция check осуществляет проверку одного файлового пути на корректную длину и на то, что все символы корректны.

```
int check(char *argument)
{
    if (symbols_check(argument) != 0){
        printf("Некорректные символы в пути");
        return -1;
    }
    if (slen(argument) > MAX_PATH){
        printf("Превышена максимальная длинна пути");
        return -1;
    }
    return 0;
}
```

Функция process обрабатывает один путь и выполняет все необходимые действия для преобразования из формата Cygwin сделать формат оригинального Windows.

```
char* process(char* tmp, char* result){
    if(check_cygwin(tmp)==1){
        for(int i=0;i<9;i++)
            delchar(tmp);
        s_upper(tmp);
        tmp[slen(tmp)]='+';
        assembly(tmp);
    }
    else if(check_cygwin(tmp)==2){
        for(int i=0;i<10;i++)
            delchar(tmp);
        s_upper(tmp);
        tmp[slen(tmp)]='+';
        assembly(tmp);
    }
    scat(result, tmp);
    return result;
}
```

Функция check_cygwin осуществляет проверку одного файлового пути на содержание формата cigwin.

```
int check_cygwin(char* str){
    int flag;
    char data[]="cygdrive/";
    char data2[]="/cygdrive";
    for(int i=0;i<slen(data);i++){
        if(str[i]==data[i])
            flag=1;
        else{
            flag=0;
            break;
        }
    }
    if(flag==0){
        for(int i=0;i<slen(data);i++){
            if(str[i]==data2[i])
                flag=2;
            else{
                flag=0;
                break;
            }
        }
    }
    return flag;
}
```

Функция assembly обрабатывает путь и выполняет преобразование корректного пути в оригинальной системе. Делает корректным имя диска и слэши.

```
char* assembly(char* str){
    char data[MAX_PATH];
    scpy(str,data);
    int size=slen(str);
    for(int i=0;i<size;i++){
        if(str[i]=='/'){
            str[i]='\\';
        }
        str[i+1]=data[i];
    }
    str[1]=':';
    return str;
}
```

Функция output осуществляет вывод передаваемой строки на экран.

```c
int output(char *str)
{
    if (printf("%s\n", str) != 0)
        return 0;
    else
        return -1;
}
```

В файле main реализуется функционал приложения с помощью вызова вышеописанных функций.

Выделяем память под необходимые в работе строки, считываем разделитель и путь.

```
char* delim = malloc(1);
char *paths = malloc(MAX_PATH * 12);
char *result = malloc(MAX_PATH * 12);
char *tmp = malloc(MAX_PATH);
printf("delim: ");
input(delim);
printf("paths: ");
input(paths);
tmp = stok(paths, delim);
```

Отделяем первый путь, в цикле, пока существуют пути, проверяем путь на корректность.

```
tmp = stok(paths, delim);
while(tmp != NULL){
        if(!check(tmp)){
                process(tmp,result);
        }
        else{
                return -1;
        }
        tmp = stok(NULL,delim);
}
```

Выводим новые пути, освобождаем выделенную память

```
    printf("new paths: %s\n",result);
    free(delim);
    free(paths);
    free(tmp);
    return 0;
```

ПРИЛОЖЕНИЕ

main.c

```c
#include <stdio.h>
#include <stdlib.h>

#include "lib.h"
#include "strings.h"

int main(){
        char* delim = malloc(1);
        char *paths = malloc(MAX_PATH * 12);
        char *result = malloc(MAX_PATH * 12);
        char *tmp = malloc(MAX_PATH);
        printf("delim: ");
        input(delim);
        printf("paths: ");
        input(paths);
        tmp = stok(paths, delim);
        while(tmp != NULL){
                if(!check(tmp)){
                        process(tmp,result);
                }
                else{
                        return -1;
                }
                tmp = stok(NULL,delim);
        }
        printf("new paths: %s\n",result);
        free(delim);
        free(paths);
        free(tmp);
        return 0;
}
```

lib.h

```
#pragma once

#define MAX_PATH 260

char *input(char *argument);
int check(char *argument);
int output(char *str);
int symbols_check(char *argument);
char* process(char* tmp, char* result);
int check_cygwin(char* str);
char* assembly(char* str1);
```

lib.c

```
#include <stdio.h>
#include <stdlib.h>

#include "lib.h"
#include "strings.h"

char *input(char *argument)
{
    scanf("%s", argument);
    return argument;
}

int symbols_check(char *argument)
{
    for (char *a = argument; *a != '\0'; a++)
    {
        if ((*a == '\\')||(*a == ':')||(*a == '*')||(*a == '?')||(*a
== '<')||(*a == '>')||(*a == '|')||(*a == '«'))
            return -1;
    }
    return 0;
}

int check(char *argument)
{
    if (symbols_check(argument) != 0){
        printf("Некорректные символы в пути");
        return -1;
    }
    if (slen(argument) > MAX_PATH){
        printf("Превышена максимальная длинна пути");
        return -1;
    }
    return 0;
}

int output(char *str)
{
```

```
39      if (printf("%s\n", str) != 0)
40          return 0;
41      else
42          return -1;
43  }
44
45  char* process(char* tmp, char* result){
46      if(check_cygwin(tmp)==1){
47          for(int i=0;i<9;i++)
48              delchar(tmp);
49          s_upper(tmp);
50          tmp[slen(tmp)]='+';
51          assembly(tmp);
52      }
53      else if(check_cygwin(tmp)==2){
54          for(int i=0;i<10;i++)
55              delchar(tmp);
56          s_upper(tmp);
57          tmp[slen(tmp)]='+';
58          assembly(tmp);
59      }
60      scat(result, tmp);
61      return result;
62  }
63
64  int check_cygwin(char* str){
65      int flag;
66      char data[]="cygdrive/";
67      char data2[]="/cygdrive";
68      for(int i=0;i<slen(data);i++){
69          if(str[i]==data[i])
70              flag=1;
71          else{
72              flag=0;
73              break;
74          }
75      }
76      if(flag==0){
77          for(int i=0;i<slen(data);i++){
78              if(str[i]==data2[i])
79                  flag=2;
80              else{
81                  flag=0;
82                  break;
83              }
84          }
85      }
86      return flag;
87  }
88
89  char* assembly(char* str){
90      char data[MAX_PATH];
91      scpy(str,data);
92      int size=slen(str);
93      for(int i=0;i<size;i++){
94          if(str[i]=='/'){
95              str[i]='\\';
96          }
97          str[i+1]=data[i];
```

```
98      }
```

strings.h

```
1   #pragma once
2
3   #include <stdio.h>
4
5   #define MAX_PATH 260
6
7   size_t slen(char *str);
8   int scmp(char *str1, char *str2);
9   char *scpy(char *str1, char *str2);
10  char* stok(char *str, char *delim);
11  char *schr(char *str, char c);
12  int is_del(char c, char *delim);
13  char* s_upper(char* str);
14  void delchar(char *str);
15  char *scat(char *str1, char *str2);
```

string.c

```
1   #include <stdio.h>
2
3   #include "strings.h"
4
5   size_t slen(char *str)//+
6   {
7       size_t len = 0;
8       for (int i = 0; str[i] != '\0'; i++)
```

```
 9          len++;
10      return len;
11  }
12
13  char* stok(char *str, char *delim)//+
14  {
15      static char *p;
16      if (!str)
17      {
18          str = p;
19      }
20      if (!str)
21      {
22          return NULL;
23      }
24      while (1)
25      {
26          if (is_del(*str, delim))
27          {
28              str++;
29              continue;
30          }
31          if (*str == '\0')
32          {
33              return NULL;
34          }
35          break;
36      }
37      char *ret = str;
38      while (1)
39      {
40          if (*str == '\0')
41          {
42              p = str;
43              return ret;
44          }
45          if (is_del(*str, delim))
46          {
47              *str = '\0';
48              p = str + 1;
49              return ret;
50          }
51          str++;
52      }
53  }
54
55  int scmp(char *str1, char *str2)//+
56  {
57      while (*str1)
58      {
59          if (*str1 != *str2)
60              break;
61          str1++;
62          str2++;
63      }
64      if((*str1 - *str2)>0)
65          return 1;
66      else if((*str1 - *str2)<0)
67          return -1;
```

```
68      return 0;
69  }
70
71  char *scpy(char *str1, char *str2)//+
72  {
73      if (str1 == NULL)
74      {
75          return NULL;
76      }
77      int len = slen(str1);
78      for(int i = 0; i < len; i++){
79          str2[i] = str1[i];
80      }
81      str2[len] = '\0';
82      return str2;
83  }
84
85  char *schr(char *str, char c)//+
86  {
87      while (*str != c && *str != '\0')
88          str++;
89      if (*str == c)
90          return str;
91      else
92          return NULL;
93  }
94
95  int is_del(char c, char *delim)
96  {
97      while (*delim != '\0')
98      {
99          if (c == *delim)
100             return 1;
101         delim++;
102     }
103     return 0;
104 }
105
106 char* s_upper(char* str){
107     str[0]=str[0]-32;
108     return str;
109 }
110
111 void delchar(char *str){
112     for(int i=0;i<slen(str);++i)
113         str[i]=str[i+1];
114 }
115
116 char *scat(char *str1, char *str2)
117 {
118     char *ptr = str1 + slen(str1);
119     while (*str2 != '\0')
120     {
121         *ptr++ = *str2++;
122     }
123     *ptr = '\0';
124     return str1;
125 }
```