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. ИС-241
«18» мая 2023 г. _____ /Дмитрюк В.В./

Проверил:
ст. преп Кафедры ВС
«19» мая 2023 г. _____ /Фульман В.О./

Оценка «_____»

Новосибирск 2023

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

Реализовать программу, преобразующую все Windows-пути формата Cygwin к оригинальным Windows-путям.

Вход:

```
delim: +
paths: cygdrive/c/Windows/system32+/cygdrive/e/Distrib/msoffice.exe+/home/alex/prog/lab4.c
```

Выход:

```
new paths: C:\Windows\system32+E:\Distrib\msoffice.exe+/home/alex/prog/lab4.c
```

ВЫПОЛНЕНИЕ РАБОТЫ

Для выполнения работы были реализованы следующие функции работы со строками:

- size_t slen(const char *str);
  Нахождение длины строки.
  - Принимает:
    - строку *str
  - Возвращает:
    - длину строки *str.

- int stok(char *string, const char delim, char *ptr[]);
  Разделение строки по известному разделителю на несколько строк. Исходная строка меняется.
  - Принимает:
    - строку *str;
    - разделитель delim;
    - пустой массив строк *ptr[].
  - Возвращает:
    - количество строк, которые были записаны в *ptr[].

- int suntok(char str[], char delim, char *ptr[], int cnt);
  Обращение действия stok.
  - Принимает:
    - строку str[];
    - разделитель delim;
    - массив строк после разделения *ptr[];
    - количество строк cnt;
  - Возвращает:
    - количество подставленных обратно разделителей.

- size_t sspn (const char *str, const char *sym);
  Проверка наличия в строке любого из определённых символов.
  - Принимает:
    - строку *str,
    - строку *sym.
  - Возвращает:
    - индекс первого символа в *str, встречающегося в строке *sym.

- int scmp(const char * string1, const char * string2);
  Сравнение двух строк.
  - Принимает:
    - строки *string1, *string2.
  - Возвращает:
    - ненулевое значение, если строки разные.
    - 0, если строки совпадают.

- int sncmp(const char * string1, const char * string2, int n);
  Сравнение n первых символов двух строк.
  - Принимает:
    - строки *string1, string2;
    - число n.
  - Возвращает:
    - ненулевое значение, если первые n символов в строках отличаются;
    - 0, если первые n символов в строках совпадают.

- char *scpy (char *dst, const char *src);
  Копирование всего содержимого строки *src в строку *dst.
  - Принимает:
    - строки *dst, *src;
  - Возвращает:
    - указатель на *dst.

- char to_upper(char c);
  Перевод символа c в верхний регистр.
  - Принимает:
    - символ c.
  - Возвращает:
    - переведённый в верхний регистр символ c, если c — строчная буква;
    - символ c без изменений в любом другом случае.

- int sconcat(char* str1, char* str2);
  Конкатенация двух строк.
  - Принимает:
    - строки *str1, *str2.
  - Возвращает:
    - длину *str1.

- void sclean(char* in);
  Очистка входной строки.
  - Принимает:
    - строку str1.

Все функции интерфейса (ввод, вывод) за исключением вывода результирующей строки реализованы в main.c.
Функция вывода результирующей строки «output» реализована в файле cyg2win.c.
Исходная задача была разбита на четыре подзадачи:

- ввод данных — функция input(...), при помощи которой считываются разделитель и разделяемая строка с путями;
- проверка корректности данных — функция check(…), проверяющая допустимость длины отдельных путей внутри входной строки (<= 260 символов) и наличие в ней запрещённых символов Windows (\/:*?«»<>|);
- обработка — функция process(…), выполняющая замену путей формата Cygwin во входной строке на пути формата Windows;
- вывод — функция output(), выводящая изменённую строку.

В эмуляторе Linux-окружения Cygwin лес деревьев-разделов дисков Windows преобразуется к дереву — все корни подключаются к заранее определённому каталогу /cygdrive. Таким образом, путь C:\akvalazy-koldunja.mp3 приобретает вид /cygdrive/c/akvalazy-koldunja.mp3.
Реализованная в ходе работы программа выполняет обратное преобразование.
Стоит заметить, что для путей, не содержащих корень /cygdrive/ или cygdrive/, преобразование не выполняется (см. Задание)

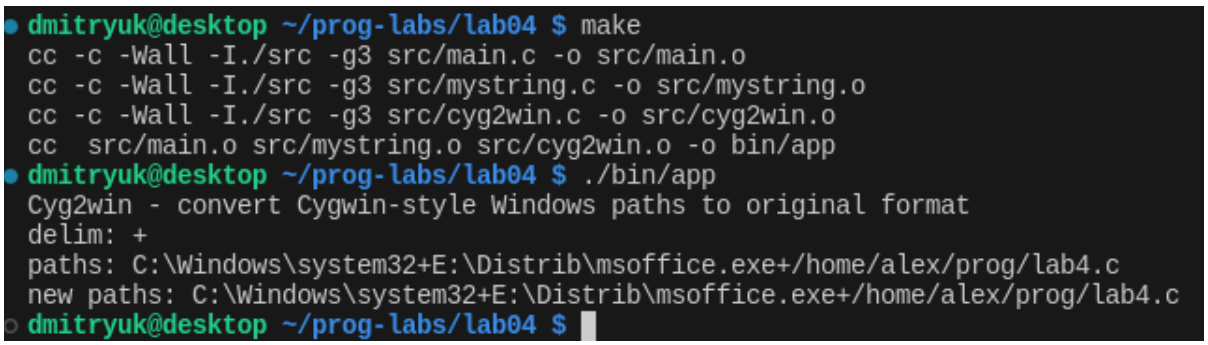

Рисунок 1. Демонстрация компилируемости и работы программы

ПРИЛОЖЕНИЕ

Исходный код с комментариями;

main.c

```
#include <cyg2win.h>

// maxlen - сколько символов может быть
// не забыть про:
// - признак конца строки
// - канарейку от переполнения
// - грязный хак для удаления переноса строки
int input(char* str, int maxlen){
        fgets(str, maxlen+3, stdin); //
        if (slen(str) >= maxlen+2)
                return -1;
        str[slen(str)-1] = '\0';
        return 0;
}

// returns index+1by of the  first invalid symbol

// this is super broken
int check(char* delim, char* paths){
        // Do both strings even exist?
        if (delim == NULL || paths == NULL) return -1;
        char* ptr[MAX_PATH_LEN];
        int errsym_index;
        // Are there any individual paths longer than 260 symbols?
        int cnt = stok(paths, *delim, ptr);
        for (int i = 0; i < cnt; i++) {
                if (slen(ptr[i]) > MAX_PATH_LEN) return i+1;
                // Does it contain forbidden symbols?
                int forbidden_pos = sspn(ptr[i], FORBIDDEN_SYMBOLS);
                if (forbidden_pos) return -(errsym_index+forbidden_pos);
                errsym_index += slen(ptr[i]);
        }
        suntok(paths, *delim, ptr, cnt);
        return 0;
}

int process(char* delim, char* pathstr){
        char* newpathst = calloc(MAX_PATHSTR_SIZE, sizeof(char));
        char* ptr[MAX_PATH_LEN];
        int cnt = stok(pathstr, *delim, ptr);
        int skiplen;
        for(int i = 0; i < cnt; i++) {
                char driveletter = '\0';
                if (!sncmp(ptr[i], "cygdrive/", 9)){
                        driveletter = ptr[i][9];
                        skiplen = 11;
                }
                else if (!sncmp(ptr[i], "/cygdrive/", 10)){
                        driveletter = ptr[i][10];
                        skiplen = 12;
                }
```

```
52                  if (driveletter) {
53                          char newstr[MAX_PATH_LEN] = {'\0'};
54                          newstr[0] = to_upper(driveletter);
55                          newstr[1] = ':';
56                          newstr[2] = '\\';
57                          for(int k = skiplen, j = 0; k < slen(ptr[i]); k++) {
58                                  if (ptr[i][k] == '/') newstr[j+3] = '\\';
59                                  else newstr[j+3] = ptr[i][k];
60                                  j++;
61                          }
62                          sconcat(newpathst, newstr);
63                          sconcat(newpathst, delim);
64                  }
65                  else{
66                          sconcat(newpathst, ptr[i]);
67                          sconcat(newpathst, delim);
68                  }
69          }
70          suntok(pathstr, delim[0], ptr, cnt);
71          newpathst[slen(newpathst)-1] = '\0';
72          sclean(pathstr);
73          scpy(pathstr, newpathst);
74          return 0;
75  }
76
77  int output(char* pathstr){
78          printf("new paths: %s\n", pathstr);
79          return 0;
80  }
```

cyg2win.h

```
 1 #include <mystring.h>
 2 #include <stdio.h>
 3 #include <errno.h>
 4 #include <stdlib.h>
 5 #define MAX_DELIM_SIZE 1
 6 #define MAX_PATHSTR_SIZE 8192
 7 #define MAX_PATH_LEN 260
 8 #define FORBIDDEN_SYMBOLS ":\\*?«<\">|"
 9 int input(char* str, int maxlen);
10 int check(char* delim, char* pathstr);
11 int process(char* delim, char* pathstr);
12 int output(char* pathstr);
```

cyg2win.c

```
1 #include <cyg2win.h>
2
3 // maxlen - сколько символов может быть
4 // не забыть про:
5 // - признак конца строки
6 // - канарейку от переполнения
7 // - грязный хак для удаления переноса строки
8 int input(char* str, int maxlen){
```

```
 9         fgets(str, maxlen+3, stdin); //
10         if (slen(str) >= maxlen+2)
11                 return -1;
12         str[slen(str)-1] = '\0';
13         return 0;
14 }
15
16 // returns index+1by of the  first invalid symbol
17
18 // this is super broken
19 int check(char* delim, char* paths){
20         // Do both strings even exist?
21         if (delim == NULL || paths == NULL) return -1;
22         char* ptr[MAX_PATH_LEN];
23         int errsym_index;
24         // Are there any individual paths longer than 260 symbols?
25         int cnt = stok(paths, *delim, ptr);
26         for (int i = 0; i < cnt; i++) {
27                 if (slen(ptr[i]) > MAX_PATH_LEN) return i+1;
28                 // Does it contain forbidden symbols?
29                 int forbidden_pos = sspn(ptr[i], FORBIDDEN_SYMBOLS);
30                 if (forbidden_pos) return -(errsym_index+forbidden_pos);
31                 errsym_index += slen(ptr[i]);
32         }
33         suntok(paths, *delim, ptr, cnt);
34         return 0;
35 }
36
37 int process(char* delim, char* pathstr){
38         char* newpathst = calloc(MAX_PATHSTR_SIZE, sizeof(char));
39         char* ptr[MAX_PATH_LEN];
40         int cnt = stok(pathstr, *delim, ptr);
41         int skiplen;
42         for(int i = 0; i < cnt; i++) {
43                 char driveletter = '\0';
44                 if (!sncmp(ptr[i], "cygdrive/", 9)){
45                         driveletter = ptr[i][9];
46                         skiplen = 11;
47                 }
48                 else if (!sncmp(ptr[i], "/cygdrive/", 10)){
49                         driveletter = ptr[i][10];
50                         skiplen = 12;
51                 }
52                 if (driveletter) {
53                         char newstr[MAX_PATH_LEN] = {'\0'};
54                         newstr[0] = to_upper(driveletter);
55                         newstr[1] = ':';
56                         newstr[2] = '\\';
57                         for(int k = skiplen, j = 0; k < slen(ptr[i]); k++) {
58                                 if (ptr[i][k] == '/') newstr[j+3] = '\\';
```

```
                                else newstr[j+3] = ptr[i][k];
                                j++;
                        }
                        sconcat(newpathst, newstr);
                        sconcat(newpathst, delim);
                }
                else{
                        sconcat(newpathst, ptr[i]);
                        sconcat(newpathst, delim);
                }
        }
        suntok(pathstr, delim[0], ptr, cnt);
        newpathst[slen(newpathst)-1] = '\0';
        sclean(pathstr);
        scpy(pathstr, newpathst);
        return 0;
}

int output(char* pathstr){
        printf("new paths: %s\n", pathstr);
        return 0;
}
```

mystring.h

```
#include <stddef.h>

size_t slen(const char *str);
int stok(char *string, const char delim, char *ptr[]);
int suntok(char str[], char delim, char *ptr[], int cnt);
size_t sspn (const char *str, const char *sym);
int scmp( const char * string1, const char * string2 );
int sncmp( const char * string1, const char * string2, int n);
char *scpy (char *dst, const char *src);
char to_upper(char c);
int sconcat(char* str1, char* str2);
void sclean(char* in);
```

mystring.c

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

size_t slen(const char *str){
    int i = 0;
    while(str[i] != '\0') i++;
    return i;
```

```
 8 };
 9
10 int schr(char str[], const char subchar) {
11         for(int i = 0; i < slen(str); i++)
12                 if (str[i] == subchar)
13                         return i;
14         return -1;
15 }
16
17 // I'm not sure of it
18 int stok(char str[], const char delim, char *ptr[]) {
19     char* suf = str;
20     ptr[0] = str;
21     int i, j = 1;
22     while((i=schr(suf,delim))>=0){
23                 suf[i] = '\0';
24                 suf = suf + i + 1;
25                 ptr[j] = suf;
26                 j++;
27         }
28         return j;
29 }
30
31 int suntok(char str[], char delim, char *ptr[], int cnt) {
32     int i;
33     for(i = 1; i < cnt; i++) {
34         *(ptr[i] - 1) = delim;
35     }
36     return i;
37 }
38
39 size_t sspn(const char *str, const char *sym) {
40     size_t len = 0;
41     for(int i = 0; i < slen(str); i++){
42         size_t tlen = len;
43         for (int k = 0; k < slen(sym); k++) {
44             if (str[i] == sym[k]){
45                 len++;
46                 break;
47             }
48         }
49         if (tlen == len) return len;
50     }
51     return len;
52 }
53
54 int scmp( const char * string1, const char * string2 ){
55     int i = 0;
56     while (string1[i] == string2[i] && string1[i] != '\0' && string2[i] != '\0')
57         i++;
```

```
58      return string1[i]-string2[i];
59  }
60
61  void sclean(char* in) {
62      int k = slen(in);
63      for (int i = 0; i < k; i++)
64          in[i] = '\0';
65  }
66
67  char *scpy (char *dst, const char *src) {
68      for (int i = 0; i < slen(src); i++)
69          dst[i] = src[i];
70      return dst;
71  }
72
73  int sncmp( const char * string1, const char * string2, int n) {
74      int i = 0;
75      while (string1[i] == string2[i] && string1[i] != '\0' && string2[i] != '\0' &&
76          i++;
77      }
78      return string1[i]-string2[i];
79  }
80
81  char to_upper(char c){
82      if (c >= 'a' && c <= 'z')
83          return c - 0x20;
84      return c;
85  }
86
87  int sconcat(char* str1, char* str2){
88      for (int i = slen(str1), j = 0; i < slen(str1)+slen(str2); i++){
89          str1[i] = str2[j];
90          j++;
91      }
92      return slen(str1);
93  }
```