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-221
«16» апреля 2023 г. _____ /Слащинин М.К./

Проверил:
Старший Преподаватель Кафедры ВС
«17» апреля 2023 г. _____ /Фульман В.О./

Оценка «_____»

Новосибирск 2023

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

Вариант №6

Преобразовать все Windows-пути формата Cygwin к оригинальным Windows-путям.

ВЫПОЛНЕНИЕ РАБОТЫ

В данной работе мне необходимо было разработать прогрмму, которая принимает на вход Windows-пути формата Cygwin и преобразовывает их в обычные Windows-пути. При этом происходят проверки на корректность пути, допустимости символов, а также на запрещенные символы. Основным требованием было разбить программу на 4 функции: **input()** - функция ввода данных; **check()** - функция проверки корректности данных; **process()** - функция обработки строки; **output()** - функция вывода данных; Также необходимо было реализовать функции обработки строк из стандартной библиотеки string.h.

функция вычисления длины строки:

```
1 size_t slen(char *mass) {
2   size_t count = 0;
3   while (*mass != '\0') {
4     count++;
5     mass++;
6   }
7   return count;
8 }
```

функция разбиения строки на элементы-токены:

```
 1 int stok(char *src, char symb, char **strs) {
 2   int i = 0;
 3   int numstrs = 1;
 4   strs[i] = src;
 5   while (src[i] != '\0') {
 6     if (src[i] == symb) {
 7       src[i] = '\0';
 8       strs[numstrs] = &(src[i]) + 1;
 9       numstrs++;
10     }
11     i++;
12   }
13   return numstrs;
14 }
```

функция проверки символа на принадлежность заданному множеству символов:

```
1 char *sspn(char *mass, char element) {
2   for (; *mass != '\0'; mass++) {
3     if (*mass == element) {
4       return mass;
5     }
6   }
7   return NULL;
8 }
```

функция сравнения строк:

```
1 int scmp(char *mass, char *mass2) {
2   int len = slen(mass);
3   for (int i = 0; i < len; i++) {
4     if (mass[i] != mass2[i]) {
5       return mass[i] - mass2[i];
6     }
7   }
8   return 0;
9 }
```

функция копирования строк:

```
1 int scmp(char *mass, char *mass2) {
2   void scpy(char *mass, char *mass2) {
3   while (*mass != '\0') {
4     *mass2++ = *mass++;
5   }
6   *mass = '\0';
7 }
```

функция восстановления строки после разбиения на элементы-токены:

```
1 void unstok(char *str, char delim, char **ptr, int cnt) {
2   for (int i = 1; i < cnt; i++) {
3     *(ptr[i] - 1) = delim;
4   }
5 }
```

функция преобразования буквы из нижнего решистра в верхний:

```c
char min_to_max(char element) { return ((int)element - 32); }
```

Помимо вышеперечисленных функций мне понадобились собственные функции и структуры:

Структура path_to — хранит в себе 2 поля. 1 поле — массив типа char размером MAX_PATH = 260, 2 поле — переменная типа size_t, которая хранит длину строки. Эта структура создана для хранения пути. Для работы с этой структурой создал несколько функций:

функция инициализации структуры:

```c
path_to *path_to_init() {
  path_to *new = (path_to *)malloc(sizeof(path_to));
  if (new == NULL) {
    return NULL;
  }
  new->len = 0;
  return new;
}
```

функция добавления элемента в строку:

```c
void path_to_add(path_to *path, char value) {
  path->string[path->len] = value;
  path->len++;
}
```

функция освобождения памяти структуры:

```c
void path_to_free(path_to *path) {
  free(path);
  path = NULL;
}
```

Также мне пригодилась еще одна структура: errors_of_path; Структура имеет 2 поля: указатель на char, переменную типа enum;

```c
typedef enum PathError {
  ErrorTypeNon,
  ErrorTypeAintPath,
  ErrorTypeTooLongLenght,
  ErrorTypeBannedSymbol,
  ErrorTypeVagueSymbol,
  ErrorTypeIncorrectPath
} PathError;
```

функция инициализации структуры:

```c
errors_of_path *errors_init() {
  errors_of_path *new = (errors_of_path *)malloc(sizeof(errors_of_path));
  if (new == NULL) {
    return NULL;
  }
  return new;
}
```

функция освобождения памяти под структуру:

```c
void errors_free(errors_of_path *error) {
  free(error);
  error = NULL;
}
```

Описание основных функций и запуск тестов:

функция **input()**

```c
void input(path_to *path) {
  printf("delim: +\n");
  printf("path: ");
  char b;
  while ((b = getchar()) != '\n') {
    path_to_add(path, b);
  }
}
```

На вход функции поступает указатель на структуру path_to, а дальше циклом while через getchar() происходит добавление элементов в строку.

функция **check()**

```
 1 void check(path_to *path, errors_of_path *error) {
 2   char *ck = sspn(path->string, '/');
 3   if (ck == NULL || (slen(path->string) < 3)) {
 4     error->type_of_error = ErrorTypeAintPath;
 5     return;
 6   }
 7   if (path->len > MAX_PATH) {
 8     error->type_of_error = ErrorTypeTooLongLenght;
 9     return;
10   }
11   char *check = black_symb(path);
12   if (check != NULL) {
13     error->type_of_error = ErrorTypeBannedSymbol;
14     error->column = check;
15     return;
16   }
17   for (int i = 0; i < path->len; i++) {
18     if (!(path->string[i] >= 32 && path->string[i] <= 126)) {
19       error->type_of_error = ErrorTypeVagueSymbol;
20       error->column = &path->string[i];
21       return;
22     }
23   }
24   char *pointers_of_path[10];
25   int count_of_path = stok(path->string, '+', pointers_of_path);
26   for (int i = 0; i < count_of_path; i++) {
27     char *pointers_of_dir[10];
28     int count_of_dir = stok(pointers_of_path[i], '/', pointers_of_dir);
29     int t = 0;
30     if (slen(pointers_of_dir[t]) == 0) {
31       t++;
32     }
33     if (scmp(pointers_of_dir[t], "cygdrive") == 0) {
34       if (!((slen(pointers_of_dir[t + 1]) == 1) &&
35             (*pointers_of_dir[t + 1] >= 97 &&
36              *pointers_of_dir[t + 1] <= 122))) {
37         error->column = pointers_of_dir[t + 1];
38         error->type_of_error = ErrorTypeIncorrectPath;
39         return;
40       }
41     }
42     unstok(pointers_of_path[i], '/', pointers_of_dir, count_of_dir);
43   }
```

```
44
45    unstok(path->string, '+', pointers_of_path, count_of_path);
46    error->type_of_error = ErrorTypeNon;
47    return;
```

Функция принимает на вход указатель на структуру path_to и указатель на структуру errors_of_path. В ней происходит проверка на ряд ошибок, и если какая-то ошибка все-таки выявилась, она передается структуре errors, если же ошибок не нашлось, в errors передается информация об их отсутствии.

функция process()

```
 1 void process(path_to *path, errors_of_path *error, path_to *new) {
 2   if (error->type_of_error != 0) {
 3     return;
 4   }
 5   char *pointers_of_path[10];
 6   int count_of_path = stok(path->string, '+', pointers_of_path);
 7   for (int i = 0; i < count_of_path; i++) {
 8     int t = 0;
 9     char *pointers_of_dir[10];
10     int count_of_dir = stok(pointers_of_path[i], '/', pointers_of_dir);
11     if (slen(pointers_of_dir[0]) == 0) {
12       t++;
13     }
14     if (scmp(pointers_of_dir[t], "cygdrive") == 0) {
15       *pointers_of_dir[t + 1] = min_to_max(*pointers_of_dir[t + 1]);
16       unstok(pointers_of_path[i], 92, pointers_of_dir, count_of_dir);
17       path_to_add(new, *pointers_of_dir[t + 1]);
18       path_to_add(new, ':');
19       for (int k = 10 + t; k < slen(pointers_of_path[i]); k++) {
20         path_to_add(new, pointers_of_path[i][k]);
21       }
22       if ((i + 1) < count_of_path) {
23         path_to_add(new, '+');
24       }
25     } else {
26       unstok(pointers_of_path[i], '/', pointers_of_dir, count_of_dir);
27       for (int k = 0; k < slen(pointers_of_path[i]); k++) {
28         path_to_add(new, pointers_of_path[i][k]);
29       }
30       if ((i + 1) < count_of_path) {
31         path_to_add(new, '+');
32       }
33     }
34   }
35 }
```

В функции происходит самая главная часть программы — преобразование пути. Функция принимает на вход 2 указателя на структуру path_to(1 указатель — наш входной путь, второй указатель — итоговый путь), указатель на структуру errors_of_path.

функция output()

```
 1 void output(path_to *path, errors_of_path *error, path_to *new) {
 2   switch (error->type_of_error) {
 3   case ErrorTypeNon:
 4         if (scmp(path->string, new->string) == 0) {
 5                printf("\n");
 6                printf("Nothing to change\n");
 7         }else {
 8                printf("\n");
 9         printf("new path: %s\n", new->string);
10         }
11         break;
12   case ErrorTypeAintPath:
13     printf("\n");
14     printf("ERROR\n");
15     printf("You didn't wrote a PATH!!!\n");
16     break;
17   case ErrorTypeTooLongLenght:
18     printf("\n");
19     printf("ERROR\n");
20     printf("Your PATH is too long!!!\n");
21     printf("Your size = %lu. Max size = 260 symbols\n", path->len);
22     break;
23   case ErrorTypeBannedSymbol:
24     printf("\n");
25     printf("ERROR\n");
26     printf("Your PATH is contain symbol: '%c' from black-list!!!\n",
27            *error->column);
28     break;
29   case ErrorTypeVagueSymbol:
30     error->column[2] = '\0';
31     printf("\n");
32     printf("ERROR\n");
33     printf("Your PATH is contain NOT available symbol: '%s'\n", error->column);
34     break;
35   case ErrorTypeIncorrectPath:
36     printf("\n");
37     printf("ERROR\n");
38     printf("Incorrect PATH!!!\n");
39     printf("Inaccuracy in the element :'%s'\n", error->column);
40     break;
41   }
42 }
```

Функция предназначена для обработки ошибок и, если их нет, вывода нового пути на экран. На вход функции поступает 2 указателя на старый и новый пути, а также указатель на структуру errors.

В конце необходимо прокрыть программу тестами.

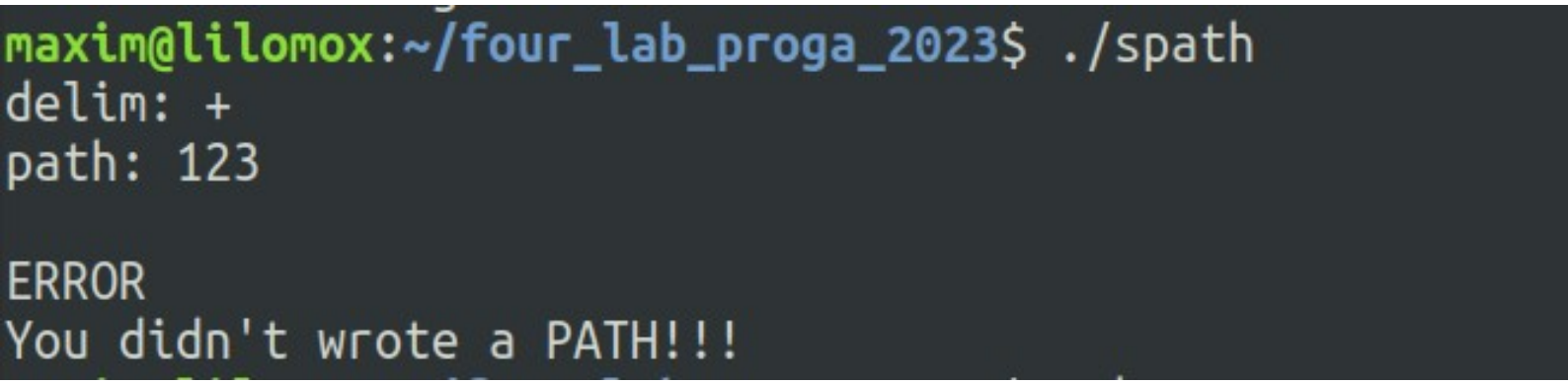
1. некорректный файловый путь.

2. превышение допустимой длины пути.

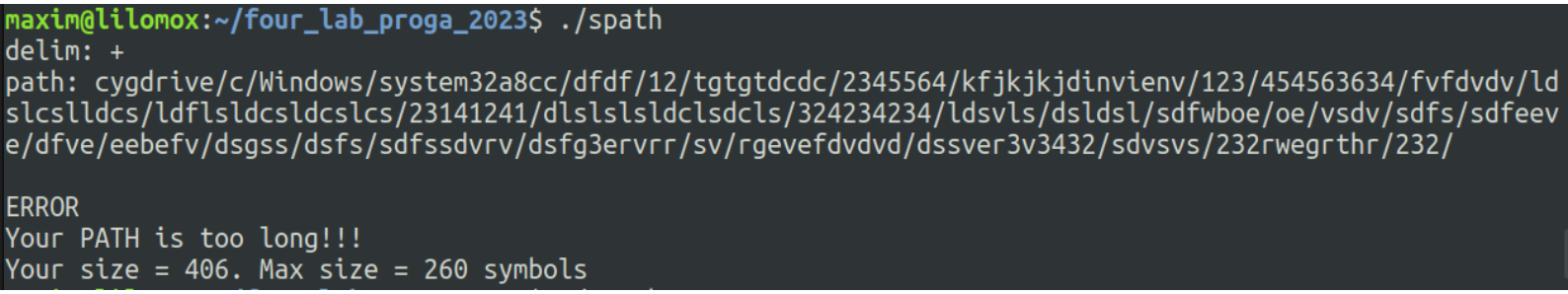
3. допустимый путь, который не удовлетворяет указанным в задании условиям.

4. допустимый путь, удовлетворяющий условиям.

1.



2.



3.

```
maxim@lilomox:~/four_lab_proga_2023$ ./spath
delim: +
path: /home/alex/prog/lab4.c

Nothing to change
```

4.

```
maxim@lilomox:~/four_lab_proga_2023$ ./spath
delim: +
path: cygdrive/c/Windows/system32+/cygdrive/e/Distrib/msoffice.exe+/home/alex/
prog/lab4.c

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

ПРИЛОЖЕНИЕ

Функция **main()** в файле **main.c**

```c
int main() {
  path_to *path;
  path_to *new;
  errors_of_path *error;

  path = path_to_init();
  new = path_to_init();
  error = errors_init();

  if (path == NULL) {
    printf("Can't allocate memory\n");
    path_to_free(new);
    errors_free(error);
    return -1;
  }
  if (new == NULL) {
    printf("Can't allocate memory\n");
    path_to_free(path);
    errors_free(error);
    return -1;
  }
  if (error == NULL) {
    printf("Can't allocate memory\n");
    path_to_free(path);
    path_to_free(new);
    return -1;
  }

  input(path);
  check(path, error);
  process(path, error, new);
  output(path, error, new);

  path_to_free(path);
  path_to_free(new);
  errors_free(error);
}
```