Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 2

По предмету «Операционные системы»
Тема: «Файлы и каталоги»

Студент Барсуков Н.М.

Группа ИУ7-66Б

Оценка (баллы) _____

Преподаватель Рязанова Наталья Юрьевна

Москва.
2020 г.

Задание:

- Структурировать исходный код программы в листинге.
- Изменить программу так, чтобы она выводила на экран дерево каталогов.
- Изменить функцию myftw() так, чтобы каждый раз, когда встречается каталог, функции lstat() передавался не полный путь к файлу, а только его имя.
  Для этого после обработки всех файлов в каталоге вызовите chdir("..").

Листинг кода

```
#include <dirent.h>
#include <limits.h>
#include <stdio.h>  //perror
#include <stdlib.h> //exit
#include <errno.h>
#include <unistd.h>    //pathconf
#include <string.h>    //strncpy
#include <sys/types.h> //stat
#include <sys/stat.h>

#define FTW_F 1    /* file, not dir */

int tree(const char *filename);
int counter(const char *pathame, const struct stat *statptr, int type);

int main(int argc, char *argv[])
{
   int ret = -1;
   if (argc != 2)
   {
      printf("Use: ftw <Stard dir>\n");
      exit(-1);
   }

   ret = tree(argv[1]);

   exit(ret);
}

typedef struct node
{
   char filepath[300];
   char foldername[300];
   int depth;
   struct node *ptrNextNode;
} node;

node *add_elem(node *head, char *filepath, char *foldername, int depth) // push&pop
{
   node *new_node = malloc(sizeof(node));
   new_node->depth = depth;
   new_node->ptrNextNode = NULL;
   strcpy(new_node->filepath, filepath);
   strcpy(new_node->foldername, foldername);

   if (head == NULL)
   {
      head = new_node;
      return head;
   }
   new_node->ptrNextNode = head;

   return new_node;
}
```

```
node *get_elem(node **head)
{
   if (*head == NULL)
      return NULL;

   node *rtrn_elem = (*head);
   (*head) = (*head)->ptrNextNode;

   return rtrn_elem;
}

void printList(node *head)
{
   for (node *iter = head; iter != NULL; iter = iter->ptrNextNode)
      printf("%s %s %d\n", iter->filepath, iter->foldername, iter->depth);
}

char *my_strcat(char *str1, char *str2)
{
   char *str_cpy = malloc(300);
   strcpy(str_cpy, str1);
   strcat(str_cpy, str2);

   return str_cpy;
}

void print_tab(int depth)
{
   for (int i = 0; i < depth; ++i)
      printf(" ├────────");
}

int tree(const char *filename)
{
   struct stat statbuf;
   struct dirent *dirp;
   DIR *dp;
   int ret = 0;
   node *head = NULL;
   node *elem = NULL;
   char *temp_str = NULL;

   do
   {
      if (head == NULL)
      {
         node *temp_node = malloc(sizeof(node));
         strcpy(temp_node->filepath, filename);
         strcpy(temp_node->foldername, "");
         temp_node->depth = 0;
         elem = temp_node;
      }
      else
         elem = get_elem(&head);

      if (elem)
```

```
        {
            if ((dp = opendir(elem->filepath)) == NULL)
            {
                counter(elem->foldername, &statbuf, FTW_DNR);
                continue;
            }

            print_tab(elem->depth - 1);

            if (counter(elem->foldername, &statbuf, FTW_D) != 0)
                continue;

            while ((dirp = readdir(dp)) != NULL)
            {
                if (strcmp(dirp->d_name, ".") == 0 || strcmp(dirp->d_name, "..") == 0)
                    continue;

                temp_str = my_strcat(my_strcat(elem->filepath, "/"), dirp->d_name);
                /*error call lstat*/
                if (lstat(temp_str, &statbuf) < 0)
                {
                    counter(dirp->d_name, &statbuf, FTW_NS);
                    continue;
                }
                free(temp_str);

                if (S_ISDIR(statbuf.st_mode) != 0)
                {
                    temp_str = my_strcat(my_strcat(elem->filepath, "/"), dirp->d_name);
                    head = add_elem(head, temp_str, dirp->d_name, elem->depth + 1);
                    free(temp_str);
                }
                else
                {
                    print_tab(elem->depth);
                    counter(dirp->d_name, &statbuf, FTW_F);
                }
            }

            if (closedir(dp) < 0)
                perror("Cannot close dir");

            if (elem)
                free(elem);
        }
    } while (head != NULL);

    return (ret);
}
int counter(const char *pathame, const struct stat *statptr, int type)
{
    if (pathame[0] == '\0')
        return 0;

    switch (type)
    {
```

```
    case ENOENT:
       printf("%s\n", pathame);
       break;
    case EBADF:
       perror("Bad file number");
       return (-1);
    case ENOMEM:
       perror("Out of memory");
       return (-1);

    case EACCES:
       perror("Permisson denied");
       return (-1);
    case EFAULT:
       perror("Bad adress");
       return (-1);

    case ENOTDIR:
       perror("Not a directory");
       return (-1);

    case EINVAL:
       perror("Invalid agruments");
       return (-1);

    case FTW_F:
       printf("%s\n", pathame);
       switch (statptr->st_mode & S_IFMT)
       {
       case S_IFREG:
          break; // simple file
       case S_IFBLK:
          break; // block device
       case S_IFCHR:
          break; // symb device
       case S_IFIFO:
          break; // FIFO chan.
       case S_IFLNK:
          break; // symb link
       case S_IFSOCK:
          break;    // socked
       case S_IFDIR: // dir
          perror("Dit type FTW_F");
          return (-1);
       }
       break;

    default:
       perror("File not dir!");
       return (-1);
    }

    return (0);
}
```

Демонстрация работы программы

Дерево каталогов: