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
Тема: вывод дерева каталогов
Дисциплина: Операционные системы

Студент Лучина Е.Д

Группа ИУ7-61Б

Преподаватель Рязанова Н.Ю.

Москва.

2020 г.

Задача
Написать программу, которая выводит дерево каталогов.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int lstat(const char *pathname, struct stat *buf);
```

Эта функция записывает информацию о файле pathname в структуру stat по указателю buf. Для этого не требуется иметь права доступа к файлу, хотя потребуются права поиска во всех каталогах, указанных в полном имени файла. Если pathname является символьной ссылкой, то lstat вернет информацию самой ссылке, а не о файле, на который она указывает.

```
struct stat {
    dev_t         st_dev;      /* у с т р о й с т в о */
    ino_t         st_ino;      /* inode */
    mode_t        st_mode;     /* р е ж и м  д о с т у п а */
    nlink_t       st_nlink;    /* к о л и ч е с т в о  ж е с т к и х
с с ы л о к */
    uid_t         st_uid;      /* и д е н т и ф и к а т о р
п о л ь з о в а т е л я - в л а д е л ь ц а */
    gid_t         st_gid;      /* и д е н т и ф и к а т о р
г р у п п ы - в л а д е л ь ц а */
    dev_t         st_rdev;     /* т и п  у с т р о й с т в а */
                               /* ( е с л и  э т о  у с т р о й с т в о ) */
    off_t         st_size;     /* о б щ и й  р а з м е р  в  б а й т а х */
    blksize_t     st_blksize;  /* р а з м е р  б л о к а
в в о д а - в ы в о д а */
                               /* в  ф а й л о в о й  с и с т е м е */
    blkcnt_t      st_blocks;   /* к о л и ч е с т в о  в ы д е л е н н ы х
б л о к о в */
    time_t        st_atime;    /* в р е м я  п о с л е д н е г о
д о с т у п а */
    time_t        st_mtime;    /* в р е м я  п о с л е д н е й
м о д и ф и к а ц и и */
    time_t        st_ctime;    /* в р е м я  п о с л е д н е г о
и з м е н е н и я */
};
```

```
#include <unistd.h>
       int chdir(const char *path);
```

Функция chdir() устанавливает в качестве текущего каталог, на который указывает параметр path. Путь может включать в себя и спецификацию диска. Каталог должен существовать. В случае успеха функция chdir() возвращает 0.

При неудаче возвращается значение -1 и устанавливается errno. Необходима в программе, чтобы использовать короткие имена.

Для определения является ли файл директорией используется макрос S_ISDIR. Возвращает ненулевое значение, если файл - директория.

```
DIR *opendir(const char *name);
```

Открывает поток каталога, соответствующий каталогу name, и возвращает указатель на этот поток. Поток устанавливается на первой записи в каталоге.

```
int closedir(DIR *dir);
```

Закрывает поток, связанный с каталогом dir. Описатель потока dir будет недоступен после вызова этой функции.

```
struct dirent *readdir(DIR *dirp);
```

Функция readdir() возвращает указатель на структуру dirent, представляющую следующую запись каталога в потоке каталога, указанного в dirp. Функция возвращает NULL по достижении последней записи в потоке каталога или если произошла ошибка.

```
struct DIR {

     struct dirent ent;

     struct _WDIR *wdirp;

};

struct dirent {

     long d_ino;

     long d_off;

     unsigned short d_reclen;

     size_t d_namlen;

     int d_type;

     char d_name[PATH_MAX+1];

};
```

Листинг программы

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <dirent.h>
```

```
#include <errno.h>
#include <stdlib.h>
#include <string.h>

void tree(char *path, int level);
char *correctcur(char *path);
```

Функция main. Принимает путь к файлу в качестве параметра командной строки. Если аргументов не два, выводит сообщение о правильном использовании. Иначе обрабатывает введенный путь функцией correctcur() и запускает вывод дерева.

```
int main(int argc, char **argv)
{
    if (argc != 2)
    {
        printf("usage: ./prog.exe path\n");
        return(-1);
    }

    char *path = correctcur(argv[1]);
    if (path != 0)
        tree(path, 0);
    return 0;
}
```

```
char *correctcur(char *path){
    printf("%s\n", path);
    if ((path[0] == '.' && path[1] == '.') || path[0] == '/')
    {
        if (chdir(path) == -1)
        {
            printf("%s: %s\n", path, strerror(errno));
            return (0);
        }
        path[0] = '.';
        path[1] = '\0';
    }
    return path;
}
```

Рекурсивная функция вывода дерева каталогов. Получает информацию о файле. Если функция lstat возвращает -1 - тупик рекурсии. Если файл не является директорией - тупик рекурсии. Иначе (если файл - директория) открываем директорию, делаем ее текущей, поочередно читаем ее содержимое, возвращаемся в родительскую директорию, закрываем папку. В момент последовательного просмотра содержимого,

печатаем сдвиг и имя файла. Если это не “.” или “..”, запускаем рекурсию, увеличивая уровень сдвига.

```
void tree(char *path, int level)
{
    struct stat file_info;

    if (lstat(path, &file_info) == -1)
    {
        printf("%s: %s\n", path, strerror(errno));
        return;
    }

    if (S_ISDIR(file_info.st_mode))
    {
        struct dirent *dirp;
        DIR *dp;

        if ((dp = opendir(path)) == 0)
        {
            printf("%s: %s\n", path, strerror(errno));
            return;
        }
        chdir(path);
        while ((dirp = readdir(dp)) != NULL) {
            for (int i = 0; i <= level; i++)
                printf("|  ");
            printf("%s\n", dirp->d_name);
            if (strcmp(dirp->d_name, ".") != 0 &&
                strcmp(dirp->d_name, "..") != 0)
                    tree(dirp->d_name, level + 1);
        }
        chdir("..");
        closedir(dp);
    }
    return;
}
```

Примеры работы:

```
myOSlabs/lab_02 $ ./a.out
usage: ./prog.exe path
myOSlabs/lab_02 $
```

```
myOSlabs/lab_02 $ ./a.out kdsjf
kdsjf
kdsjf: No such file or directory
myOSlabs/lab_02 $ ./a.out ../../dfksjd
../../dfksjd
../../dfksjd: No such file or directory
myOSlabs/lab_02 $ ./a.out /kjkj
/kjkj
/kjkj: No such file or directory
myOSlabs/lab_02 $
```

```
myOSlabs/lab_02 $ ./a.out dir2
dir2
|   .
|   ..
myOSlabs/lab_02 $
```

```
myOSlabs/lab_02 $ ./a.out /
/
|   .
|   ..
|   bin
|   |   .
|   |   ..
|   |   bash
|   |   btrfs
|   |   btrfs-calc-size
|   |   btrfs-convert
|   |   btrfs-debug-tree
|   |   btrfs-find-root
|   |   btrfs-image
|   |   btrfs-map-logical
|   |   btrfs-select-super
|   |   btrfs-show-super
|   |   btrfs-zero-log
|   |   btrfsck
```

```
myOSlabs/lab_02 $
myOSlabs/lab_02 $ ./a.out .
.
|   .
|   ..
|   a.out
|   dir1
|   |   .
|   |   ..
|   |   dir12
|   |   |   .
|   |   |   ..
|   |   |   dir123
|   |   |   |   .
|   |   |   |   ..
|   |   |   |   foo
|   |   file1
|   |   file2
|   dir2
|   |   .
|   |   ..
|   recursion.c
myOSlabs/lab_02 $
```

```
myOSlabs/lab_02 $
myOSlabs/lab_02 $ ./a.out ..
..
|   .
|   ..
|   lab_01
|   |   .
|   |   ..
|   |   a.out
|   |   main.c
|   |   trylog.c
|   |   trylog.exe
|   lab_02
|   |   .
|   |   ..
|   |   a.out
|   |   dir1
|   |   |   .
|   |   |   ..
|   |   |   dir12
|   |   |   |   .
|   |   |   |   ..
|   |   |   |   dir123
|   |   |   |   |   .
|   |   |   |   |   ..
|   |   |   |   |   foo
|   |   |   file1
|   |   |   file2
|   |   dir2
|   |   |   .
|   |   |   ..
|   |   recursion.c
myOSlabs/lab_02 $
```