|  |  |
| --- | --- |
|  | **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:

