МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ НЭТИ | Факультет прикладной математики и информатики

Кафедра прикладной математики

Лабораторная работа № 1

по дисциплине «Управление ресурсами в вычислительных системах»

ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ УПРАВЛЕНИЯ СИСТЕМНЫМИ РЕСУРСАМИ. ФАЙЛОВАЯ СИСТЕМА ОС UNIX.

Бригада 7	ГРУШЕВ АНДРЕЙ
Группа ПМ-05	БОЛДЫРЕВ СЕРГЕЙ
Вариант 7	ХАБАРОВА АНАСТАСИЯ

Преподаватели	СТАСЫШИН ВЛАДИМИР МИХАЙЛОВИЧ
	СИВАК МАРИЯ АЛЕКСЕЕВНА

Новосибирск, 2023

Условие

Программа выводит содержимое каталога, имя которого указано параметром программы. При выводе сначала перечисляются имена каталогов, а затем в алфавитном порядке имена файлов с указанием их длин, даты последнего изменения и числа ссылок на них.

Используемые программные средства

Языка **C**:

DIR* opendir(char* path) – функция открывает каталог соответствующий пути path. Возвращает указатель на дескриптор типа DIR* или NULL в случае ошибки.

closedir(DIR* dirptr) – функция закрывает каталог, по дескриптору dirptr.

struct dirent* readdir(DIR* dir) – функция последовательно считывает элемент каталога. Возвращает указатель дескриптор каталога DIR* или в случае ошибки.

stat(char* d_name, struct stat* buf) – помещает информацию о файле d_name в переменную на которую указывает buf.
Макросы **S_ISREG(mode_t st_mode)** и **S_ISDIR(mode_t st_mode)** проверяют, является ли файл обычным файлом и каталогом соответственно, в зависимости от переданного в качестве аргумента st_mode структуры stat.

char* ctime (const time_t *ttime) - функция преобразует время в секундах, истекшее с 0 часов 1 января 1970 года и возвращает текстовую строку, с учетом часового пояса.

printf(const char* format, …) – форматированный вывод в файл стандартного вывода.

fprintf(FILE* stream, const char *format, …) – форматированный вывод в файл на который указывает stream.

sprintf(char* str, const char* format, …) – форматированный вывод в символьную строку на которую указывает str.

void exit(int status) – функция приводит к обычному завершению программы. Стандарт C описывает два определения EXIT_SUCCESS и EXIT_FAILURE, которые могут быть переданы функции для обозначения соответственно успешного и неуспешного завершения.

Языка **Shell**:

printf FORMAT [VALUES] - вывод VALUES согласно указанному FORMAT.

basename FILE - вывод имени файла FILE, удаляя все существующие в начале каталоги.

stat –format=FORMAT FILE – выводит содержимое структуры stat файла FILE.

Алгоритм решения

Для программы на языке **C:**

1. Считываем последовательно записи заданного каталога, при этом имена каталогов выводим сразу, а нужную информацию о файлах записываем в массив вспомогательных структур.
2. Сортируем массив вспомогательных структур по именам файлов по алфавиту.
3. Выводим содержимое массива вспомогательных структур.

Для скрипта на языке **Shell:**

1. Считываем последовательно записи заданного каталога и выводим имена каталогов.
2. Считываем последовательно записи заданного каталога и выводом имена файлов и нужную информацию о них.

Спецификация

Код программы расположен на сервере НГТУ в директории **/home/NSTU/pmi-b0507/upres/lab1**.
Файл с кодом на языке **C** – **main.c**; на языке **Shell** – **main.sh**.

Для получения исполняемого файла необходимо находясь в директории с файлом **main.c** выполнить команду:
gcc main.c -o [имя_исполняемого_файла],
либо воспользоваться make-файлом при помощи команды:
make main,
которая создаст исполняемый файл **main.o**.

Запуск программы происходит при помощи команды:
./[имя_исполняемого_файла] [имя_каталога-аргумента],
например:
./main.o test или **./main.sh test**.

Формат вывода результата:

```
Directory – [имя_директории]
…

[имя_файла]
Size – [размер_файла]
Time – [время_последнего изменения_файла]
Links – [количество_жёстких_ссылок_на_файл]

…
```

Перечень тестов

Для тестирования программ была подготовлена расположенная в том же каталоге директория **test** следующего содержания:

```
-rw-r--r--. 1 pmi-b0507 пользователи домена   62 Feb 19 17:55 abcdef.txt
-rw-r--r--. 2 pmi-b0507 пользователи домена    0 Feb 15 04:50 abcdlink
-rw-r--r--. 2 pmi-b0507 пользователи домена    0 Feb 15 04:50 abcd.txt
-rw-r--r--. 1 pmi-b0507 пользователи домена    0 Feb 15 04:50 file1.txt
-rw-r--r--. 1 pmi-b0507 пользователи домена   21 Feb 15 22:04 file.txt
drwxr-xr-x. 2 pmi-b0507 пользователи домена 4096 Feb 15 04:48 test1
drwxr-xr-x. 2 pmi-b0507 пользователи домена 4096 Feb 15 04:48 test2
drwxr-xr-x. 2 pmi-b0507 пользователи домена 4096 Feb 15 04:48 test3
```

Для программы на языке **C:**

№	Входные данные	Назначение	Результаты работы программы
1	`./main.o`	Запуск программы с недостаточным количеством параметров.	`Error! Wrong number of arguments (expected 1, given 0).`
2	`./main.o test 2`	Запуск программы с излишним количеством параметров.	`Error! Wrong number of arguments (expected 1, given 2).`
3	`./main.o test/abcd.txt`	Указанный параметр не является директорией.	`Error! test/abcd.txt cannot be opened.`
4	`./main.o test/asdzxc`	Указанная директория не существует.	`Error! test/asdzxc cannot be opened.`

5	`./main.o test`	Запуск программы с правильным количеством параметров (путь до директории относительный).	`Directory - test2`
			`Directory - test1`
			`Directory - test3`

			abcd.txt Size - 0 bytes Time - Wed Feb 15 04:50:22 2023 Links - 2 abcdef.txt Size - 62 bytes Time - Sun Feb 19 17:55:36 2023 Links - 1 abcdlink Size - 0 bytes Time - Wed Feb 15 04:50:22 2023 Links - 2 file.txt Size - 21 bytes Time - Wed Feb 15 22:04:07 2023 Links - 1 file1.txt Size - 0 bytes Time - Wed Feb 15 04:50:26 2023 Links - 1
6	./main.o /home/NSTU/pmi-b0507/upres/lab1/test	Запуск программы с правильным количеством параметров (путь до директории абсолютный).	Directory - test2 Directory - test1 Directory - test3 abcd.txt Size - 0 bytes Time - Wed Feb 15 04:50:22 2023 Links - 2 abcdef.txt Size - 62 bytes Time - Sun Feb 19 17:55:36 2023 Links - 1 abcdlink Size - 0 bytes Time - Wed Feb 15 04:50:22 2023 Links - 2 file.txt Size - 21 bytes Time - Wed Feb 15 22:04:07 2023 Links - 1 file1.txt Size - 0 bytes Time - Wed Feb 15 04:50:26 2023 Links - 1

Для скрипта на языке **Shell:**

№	Входные данные	Назначение	Результаты работы программы
1	./main.sh	Запуск программы с недостаточным количеством параметров.	Error! Wrong number of arguments (expected 1, given 0).
2	./main.sh test 2	Запуск программы с излишним количеством параметров.	Error! Wrong number of arguments (expected 1, given 2).
3	./main.sh test/abcd.txt	Указанный параметр не является директорией.	Error! test/abcd.txt cannot be opened.
4	./main.sh test/asdzxc	Указанная директория не существует.	Error! test/asdzxc cannot be opened.
5	./main.sh test	Запуск программы с правильным количеством параметров (путь до директории относительный).	Directory - test1 Directory - test2 Directory - test3

			File - abcdef.txt Size - 62 bytes Time - 2023-02-19 17:55:36.560576198 +0700 Links - 1 File - abcdlink Size - 0 bytes Time - 2023-02-15 04:50:22.522981302 +0700 Links - 2 File - abcd.txt Size - 0 bytes Time - 2023-02-15 04:50:22.522981302 +0700 Links - 2 File - file1.txt Size - 0 bytes Time - 2023-02-15 04:50:26.324060580 +0700 Links - 1 File - file.txt Size - 21 bytes Time - 2023-02-15 22:04:07.365536419 +0700 Links - 1
6	./main.sh /home/NSTU/pmi-b0507/up-res/lab1/test	Запуск программы с правильным количеством параметров (путь до директории абсолютный).	Directory - test1 Directory - test2 Directory - test3 File - abcdef.txt Size - 62 bytes Time - 2023-02-19 17:55:36.560576198 +0700 Links - 1 File - abcdlink Size - 0 bytes Time - 2023-02-15 04:50:22.522981302 +0700 Links - 2 File - abcd.txt Size - 0 bytes Time - 2023-02-15 04:50:22.522981302 +0700 Links - 2 File - file1.txt Size - 0 bytes Time - 2023-02-15 04:50:26.324060580 +0700 Links - 1 File - file.txt Size - 21 bytes Time - 2023-02-15 22:04:07.365536419 +0700 Links - 1

Make-файлы

Файл **makefile:**

```
main: main.c
    gcc main.c -o main.o

clean:
    rm main.o
```

Листинг программы

Код программы на языке **C**:

```c
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>

/**
 * Вспомогательная структура, которая содержит
 * нужную по заданию информацию о файле.
**/
struct file_info
{
   char* name; //Имя файла.
   off_t st_size; // Размер в байтах.
   time_t time; // Время последней модификации.
   nlink_t st_nlink; // Количество жёстких ссылок.
};

/**
 * Функция сортирует массив вспомогательных структур по имени
 * в алфавитном порядке при помощи сортировки пузырьком.
 * @param array Массив, который нужно отсортировать.
 * @param size Размер массива.
 **/
void sortInfos(struct file_info array[], int size)
{
   struct file_info tmp;
   int i, j;

   for (i = 1; i < size; i++)
      for (j = 0; j < size - i; j++)
         if (strcmp(array[j].name, array[j + 1].name) > 0)
         {
            tmp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = tmp;
         }
}

int main(int argc, char** argv)
{
   DIR* d;

   struct dirent* dirElement;
   struct stat buff;
```

```
    char path[50];
    struct file_info regularFiles[50];

    int filesCount = 0;
    int i;

    if (argc != 2)
    {
        fprintf(stderr, "Error! Wrong number of arguments (expected 1, given %d).\n", --
argc);
        exit(EXIT_FAILURE);
    }

    d = opendir(argv[1]);
    if (!d)
    {
        fprintf(stderr, "Error! %s cannot be opened.\n", argv[1]);
        closedir(d);
        exit(EXIT_FAILURE);
    }

    while ((dirElement = readdir(d)) != NULL) // Чтение файлов из заданной директории.
        if (dirElement->d_name[0] != '.') //Пропуск скрытых файлов.
        {
            sprintf(path, "%s/%s", argv[1], dirElement->d_name); //Формируем путь к файлу,
для функции stat().
            stat(path, &buff); //Получаем информацию о файле.
            if (S_ISDIR(buff.st_mode)) //Проверка является ли файл директорией.
                printf("Directory - %s\n", dirElement->d_name); //Печать имени файла.
            else if (S_ISREG(buff.st_mode)) //Проверка является ли файл директорией.
            {
                //Заносим информацию о файле в массив.
                regularFiles[filesCount].name = dirElement->d_name;
                regularFiles[filesCount].st_size = buff.st_size;
                regularFiles[filesCount].time = buff.st_mtime;
                regularFiles[filesCount].st_nlink = buff.st_nlink;

                filesCount++;
            }
        }

    sortInfos(regularFiles, filesCount); //Сортируем массив структур по алфавиту.
    printf("\n");

    for (i = 0; i < filesCount; i++)
    {
        printf("%s\n", regularFiles[i].name);
        printf("Size - %ld bytes\nTime - %sLinks - %ju\n\n",
              regularFiles[i].st_size,
              ctime(&regularFiles[i].time),
              regularFiles[i].st_nlink); //Печатаем нужную информацию о файле.
    }

    closedir(d);
    return 0;
}
```

Код скрипта на языке **Shell**:

```
# !/bin/bash

if [ $# -eq 1 ]
then
    if [ -d $1 ]
    then
        if [ ${1: -1} == "/" ]
        then
            path="$1*"
        else
            path="$1/*"
        fi

        for file in $path
        do
            if [ -d $file ]
            then
                printf 'Directory - %s\n' "$(basename $file)"
            fi
        done

        printf '\n'

        for file in $path
        do
            if [ -f $file ]
            then
                printf 'File - %s\n' "$(basename $file)"
                printf '%s\n\n' "$(stat --printf="Size - %s bytes\nTime - %y\nLinks -
%h" $file)"
            fi
        done
    else
        printf 'Error! %s cannot be opened.\n' "$1"
    fi
else
    printf 'Error! Wrong number of arguments (expected 1, given %d).\n' "$#"
fi
```