**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФГБОУ ВО**

**Московский авиационный институт**

**(национальный исследовательский университет)**

Кафедра 304

Программное обеспечение средств вычислительной

техники и автоматизированных систем

**Отчет по лабораторной работе №1**

**по учебной дисциплине «Операционные системы»**

**на тему**

**«Архиватор»**

Выполнили студенты группы М3О-310Б-20:

Ратников Р.О.

Кильмишкин Н.В.

Приняли:

Титов Ю.П.

Грабовский М.Н.

Москва 2023

**Содержание**

[**Задание** 3](#_Toc128511198)

[**Листинг программы** 4](#_Toc128511199)

[**Работа программы** 13](#_Toc128511200)

# **Задание**

1. Написать программу архиватор, на вход которой передается папка с файлами, результатом работы программы является файл, объединяющий все файлы из входной директории, а также содержащий заголовок с информацией об именах файлов и их размерах.

2. Написать программу разархиватор, разделяющий архивированный файл обратно на файлы.

# **Листинг программы**

#include <unistd.h>

#include <stdio.h>

#include <dirent.h>

#include <string.h>

#include <sys/stat.h>

#include <stdlib.h>

/\* Определение размера файла и запись в bout\*/

int getFileSize(const char\* file\_name, FILE \*\*bout)

{

int file\_size = 0;

FILE\* fd = fopen(file\_name, "rb");

char byte[1];

if(fd == NULL)

{

file\_size = -1;

}

else

{

while(fread(byte,1,1,fd)==1)

{

file\_size++;

fwrite(byte,1,1,\*bout);

}

fclose(fd);

}

return file\_size;

}

// Табуляции

void tabs (int depth)

{

for (int i = 0; i < depth; i ++) printf("\t");

return;

}

int print\_info (int depth, char\* Temp3, char\* name, FILE \*\*bout)

{

tabs (depth);

printf("%s\n", Temp3);

int file\_size = getFileSize(name, bout);

tabs (depth);

printf("File size: %d bytes\n", file\_size);

tabs (depth);

return (file\_size);

}

void printdir(char \*dir, int depth, char rep[100], FILE \*\*info\_files, FILE \*\*info\_direct, FILE \*\*bout)

{

DIR \*dp;

struct dirent \*entry;

struct stat statbuf;

// Проверка на возможность открыть файл

if ((dp = opendir(dir)) == NULL)

{

// выводит результат в заданный файловый поток stream

fprintf(stderr, "cannot open directory: %s\n", dir);

return;

}

// Изменяет текущий каталог на заданный

chdir(dir);

// Пока чтение директории не закончится

while((entry = readdir(dp)) != NULL)

{

// Возвращение информации о файле

lstat(entry->d\_name, &statbuf);

//char \*Temp0 = StartDir;

char \*Temp1 = rep;

char \*Temp2 = entry->d\_name;

char Temp3[100];

strcpy(Temp3, Temp1);

strcat(Temp3, "/");

strcat(Temp3, Temp2);

// Проверка на каталог

if (S\_ISDIR(statbuf.st\_mode))

{

// Находит каталог, но игнорирует . и ..

// strcmp сравнение, если строки равны, то возвращает 0

if (strcmp(".", entry->d\_name) == 0 || strcmp("..", entry->d\_name) == 0)

{ continue; }

print\_info(depth, Temp3, entry->d\_name, bout);

printf("Type: Directory\n\n");

fprintf(\*info\_direct, "%s||", Temp3 );

/\* Рекурсивный вызов с новый отступом \*/

printdir(entry->d\_name, depth+1, Temp3, info\_files, info\_direct, bout);

}

// Для не каталога

else

{

if((strcmp(entry->d\_name, "info\_files.txt")==0) || (strcmp(entry->d\_name, "info\_direct.txt")==0) || (strcmp(entry->d\_name, "bout.txt") ==0))

{ continue; }

int file\_size = print\_info(depth, Temp3, entry->d\_name, bout);

printf("Type: File\n\n");

fprintf(\*info\_files, "%s|%d|%s||", entry->d\_name, file\_size, Temp3 );

}

}

// Изменяет текущий каталог на заданный

chdir("..");

// Закрывает заданый каталог

closedir(dp);

}

void enter\_directory(char\* dir, int mode)

{

// Дефолт для ввода

if (mode == 1)

{

printf(" Введите директорию архивации\n");

}

// Дефолт для вывода

if (mode == 0)

{

printf(" Введите директорию разархивации\n");

}

while(1)

{

printf(" По умолчанию: %s\n", dir);

printf(" 1) По умолчанию\n");

printf(" 2) Задать\n");

printf(" Выбор: ");

//cout << " Для выхода нажмите 0" << endl;

int Choice;

char Input\_dir[256];

scanf("%d", &Choice);

if(Choice == 1)

{

break;

} //if

if(Choice == 2)

{

printf(" Введите путь: ");

scanf("%s", Input\_dir);

printf("\n\n");

if (opendir(Input\_dir) == NULL)

{

printf(" cannot open directory: %s\n", Input\_dir);

printf(" Попробуйте ещё раз\n");

printf(" Для ввода по умолчанию нажмите не 2\n\n");

}//if

else

{

if((strcmp(Input\_dir, "/home") == 0) || (strcmp(Input\_dir, "/") == 0))

{

printf(" Эту папку архивировать нельзя: %s\n", Input\_dir);

printf(" Попробуйте ещё раз\n");

printf(" Для ввода по умолчанию нажмите не 2\n\n");

}//if

else

{

strcpy(dir, Input\_dir);

break;

}//else

}//else

}//if

else

{

printf(" Выбрано по умолчанию\n");

break;

}

}

printf("\n");

}

void files\_archivation(char name\_info\_direct\_bout[4][100], char\* RestOfDir, char\* StartDir)

{

char char\_info\_files[100] = "/info\_files.txt";

char char\_info\_direct[100] = "/info\_direct.txt";

char char\_bout[100] = "/bout.txt";

char char\_main[100];

strcpy(char\_main, StartDir);

strcat(char\_main, ".zippo");

for(int i = 0; i < 4;i++)

{

char Temp[256];

strcpy(Temp, RestOfDir);

if (i != 3)

{

strcat(Temp, StartDir);

}

strcat(Temp, name\_info\_direct\_bout[i]);

strcpy(name\_info\_direct\_bout[i], Temp);

}

printf("\n");

strcat(name\_info\_direct\_bout[0], char\_info\_files);

strcat(name\_info\_direct\_bout[1], char\_info\_direct);

strcat(name\_info\_direct\_bout[2], char\_bout);

strcat(name\_info\_direct\_bout[3], char\_main);

return;

}

void arhivation ()

{

// Определение директории по умолчанию

char\* dir\_default;

char Temp[100] = "/home/nikita/Документы/labs/ModelForLab1";

dir\_default = Temp;

//Выбор директории

enter\_directory(dir\_default, 1);

char \*dir = dir\_default;

char StartDir[100];

char RestOfDir[256];

strcpy(Temp, dir);

// Рабочие файлы (в конце удаляются)

FILE \*info\_files; // Хранятся названия файлов, размер, директория этого файла

FILE \*info\_direct; // Хранится директория папок

FILE \*bout; // Хранится информация файлов

// Файл архива

FILE \*main; // Итоговый файл, в котором info\_files, info\_direct, bout

// Инициализация и очистка массива

char name\_info\_direct\_bout[4][100];

for(int i = 0; i < 4; i++)

{

strcpy(name\_info\_direct\_bout[i], "");

}

printf(" Выбранная папка: %s\n", Temp);

// Определение конечной папки

char sep [2] = "/";

int count1 = 0;

int count2 = 0;

while (dir[count2] != 0)

count2++;

count1 = count2;

while (dir[count1] != \*sep)

count1--;

// Опеределение конечной папки

for(int i = 0; i < (count2-count1 + 1);i++)

{

StartDir[i] = dir[count1+i];

}

// Опеределние директории конечной папки

if (count1 != 0)

{

//printf( " dir[%d] %s\n", count1, &dir[count1]);

for(int i = 0; i < count1; i++)

{

RestOfDir[i] = dir[i];

}

RestOfDir[count1] = 0;

}

// Определение директории для рабочих файлов и файла архива

files\_archivation(name\_info\_direct\_bout, RestOfDir, StartDir );

//return;

//printf( " StartDir %s\n", StartDir);

strcat(StartDir, "(zippo)");

printf("Куда будут записываться файлы\n");

for(int i = 0; i < 4; i++)

{

printf(" %d %s\n", i, name\_info\_direct\_bout[i]);

}

printf("\n");

// Открытие файлов

info\_files = fopen(name\_info\_direct\_bout[0], "wb");

info\_direct = fopen(name\_info\_direct\_bout[1], "wb");

bout = fopen(name\_info\_direct\_bout[2], "wb");

printf("Directory аrchiving of %s\n", dir);

fprintf(info\_direct, "%s||", StartDir);

printdir(dir, 0, StartDir, &info\_files, &info\_direct, &bout);

printf("done.\n");

fprintf(info\_files, "\n");

fprintf(info\_direct, "\n");

fclose(info\_files);

fclose(info\_direct);

fclose(bout);

char byte[1]; // единичный буфер для считывания одного байта

// Открытие файлов для чтения

info\_files = fopen(name\_info\_direct\_bout[0], "rb");

info\_direct = fopen(name\_info\_direct\_bout[1], "rb");

bout = fopen(name\_info\_direct\_bout[2], "rb");

// Открытие файла для записи

main = fopen(name\_info\_direct\_bout[3], "wb");

// Можно сделать через цикл

// переписываем информацию с файлами в файл main

while(!feof(info\_files))

{

if(fread(byte,1,1,info\_files)==1) fwrite(byte,1,1,main);

}

// переписываем информацию с директориями в файл main

while(!feof(info\_direct))

{

if(fread(byte,1,1,info\_direct)==1) fwrite(byte,1,1,main);

}

// Переписываем данные в файл main

while(!feof(bout))

{

if(fread(byte,1,1,bout)==1) fwrite(byte,1,1,main);

}

// Закрытие

fclose(info\_files);

fclose(info\_direct);

fclose(bout);

// Удаление рабочих файлов

for(int i = 0; i < 3; i++)

remove(name\_info\_direct\_bout[i]);

// Закрытие файла с архивом

fclose(main);

}

void unrar ();

int main(int argc, char\* argv[])

{

while(1)

{

int key;

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf(" Введите (1|0)\n");

printf(" 1) Архивировать\n");

printf(" 2) Разархивировать\n");

printf(" 3) Выйти\n");

printf(" Выбор: ");

scanf("%d", &key);

printf("\n");

// Можно сделать кэйсы

// Печать директории и запись информации в файл

if (key == 1)

arhivation();

// функция разархивации, подаётся путь разархивации и файл с данными

if (key == 2)

unrar();

// Выход

if (key == 3)

break;

}//while

return 0;

}

void unrar ()

{

//FILE \*main = fopen("/home/nikita/Документы/main.zippo", "rb");

FILE \*main; // файл архива

char DirUnrar[256]; // Директория файла

// Попытка открыть файл

while(1)

{

// Ввод информации

printf(" Введите значение\n");

printf(" Путь до файла архива: ");

scanf("%s", DirUnrar);

//printf(" Длина строки %s: %ld\n", DirUnrar, strlen(DirUnrar));

printf("\n");

//printf(" %d\n", (strcmp(&DirUnrar[strlen(DirUnrar)-6], ".zippo")));

// Если доступ есть

if(access(DirUnrar, F\_OK) == 0)

{

// Проверка на расширение

if((strcmp(&DirUnrar[strlen(DirUnrar)-6], ".zippo")) == 0)

{

main = fopen(DirUnrar, "rb");

printf(" Файл успешно открыт\n");

break;

}

else

{

// Нужна функция

int key;

printf(" Это либо папка, либо неправильное расширение\n");

printf("\*Выйти (0)\n");

printf("\*Попробовать ещё раз (1)\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Выбор: \n");

scanf("%d", &key);

if (key == 1) {}

else {return;}

}

}

// Если доступа нет

else

{

int key;

printf(" Невозможно открыть файл\n");

printf("\*Выйти (0)\n");

printf("\*Попробовать ещё раз (1)\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Выбор: \n");

scanf("%d", &key);

if (key == 1) {}

else {return;}

}

}

main = fopen(DirUnrar, "rb");

// Создание директории

char\* dir\_default;

char Temp[100] = "/home/nikita/Документы/labs/Resforlab1";

dir\_default = Temp;

char str[10000]; // Типо буфера

char \*first\_str; // Для первой строки из main

char \*istr;

// Разделил первую строку

first\_str = fgets (str, sizeof(str), main);

char sep [4] = "|";

// Разделил на Name, Size, Path

istr = strtok (first\_str, sep);

char \*file [100][3];

int count = 0;

// Разделение на составляющие (Name, Size, Path)

while ((istr != NULL) && (istr[0] != 10) /\*&& (istr[0] != 32)\*/)

{

//cout << count << " ";

for (int i = 0; i < 3; i++)

{

file[count][i] = istr;

istr = strtok (NULL,sep);

}

count++;

}

// Разделил на директории

char str1[10000];

char \*second\_str; // Для второй строки из main

char \*istr1;

second\_str = fgets (str1, sizeof(str1), main);

istr1 = strtok (second\_str, sep);

char \*file1 [100];

int count\_dir = 0;

if(istr1 == NULL) scanf(" (istr1 == NULL)");

printf(" Читаем архив...\n");

printf(" Папочки:\n");

while ((istr1 != NULL) && (istr1[0] != 10) && (istr[0] != 32))

{

printf("%d ", count\_dir);

file1[count\_dir] = istr1;

istr1 = strtok (NULL,sep);

printf("%s\n", file1[count\_dir]);

count\_dir++;

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Totaly dir: %d\n\n", count\_dir);

printf(" Файлики:\n");

// Просто печать что получилось

for (int i = 0; i < count; i++)

{

printf("%2d", i);

printf("%22s ", file[i][0]);

printf("%7s ", file[i][1]);

printf("%55s ", file[i][2]);

printf("\n");

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Totaly files: %d\n\n", count);

// Подтверждение разархивации

printf(" Разархивировать?\n");

printf(" Для продолжения введите 1\n");

printf(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf(" Ввод:");

int k = 0;

scanf("%d", &k);

if(k == 1) {}

else {return;}

printf(" Где будет директория?\n");

enter\_directory(dir\_default,2);

char\* DirOutPut = dir\_default;

printf(" DirOutPut: %s\n", DirOutPut);

printf(" Распаковка сюда: %s\n", DirOutPut);

printf(" Создание папок...\n");

for (int i = 0; i < count\_dir; i++)

{

char \*CreateNewDir = DirOutPut;

char\* Temp = file1[i];

char PathDir[100];

// конкотенация

strcpy(PathDir, CreateNewDir);

strcat(PathDir, Temp);

printf("PathDir: %s\n", PathDir);

mkdir(PathDir, 00777);

}

printf("\n");

printf(" Запись в папки...\n");

char byte[1];

// Создание файлов и побайтовая их перепись согласно количеству файлов

for (int j = 0; j < count; j++)

{

char\* PathSet = DirOutPut;

char\* PathFile = file[j][2];

char PathRes[256];

strcpy(PathRes, PathSet);

strcat(PathRes, PathFile);

printf("PathRes: %s\n", PathRes);

int Size = atoi(file[j][1]);

//cout << "\tSize: " << Size << " bytes"<< endl;

FILE \*Temp = fopen(PathRes, "wb");

// Побайтовая перепить согласно заданному размеру

for(int i = 0; i < Size; i++)

{

if(fread(byte,1,1, main) == 1)

{

fwrite(byte,1,1,Temp);

}

}

fclose(Temp);

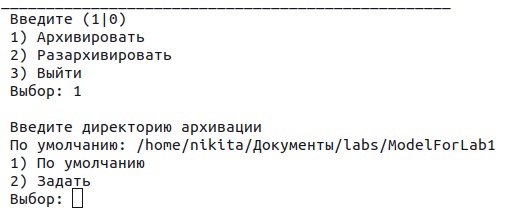
}

fclose(main);

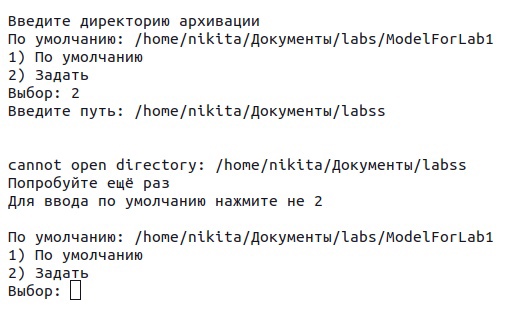
}

# **Работа программы**

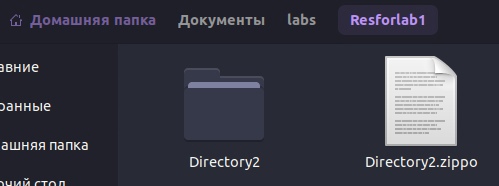
При вводе архивации в главном меню на экране появляется выбор:

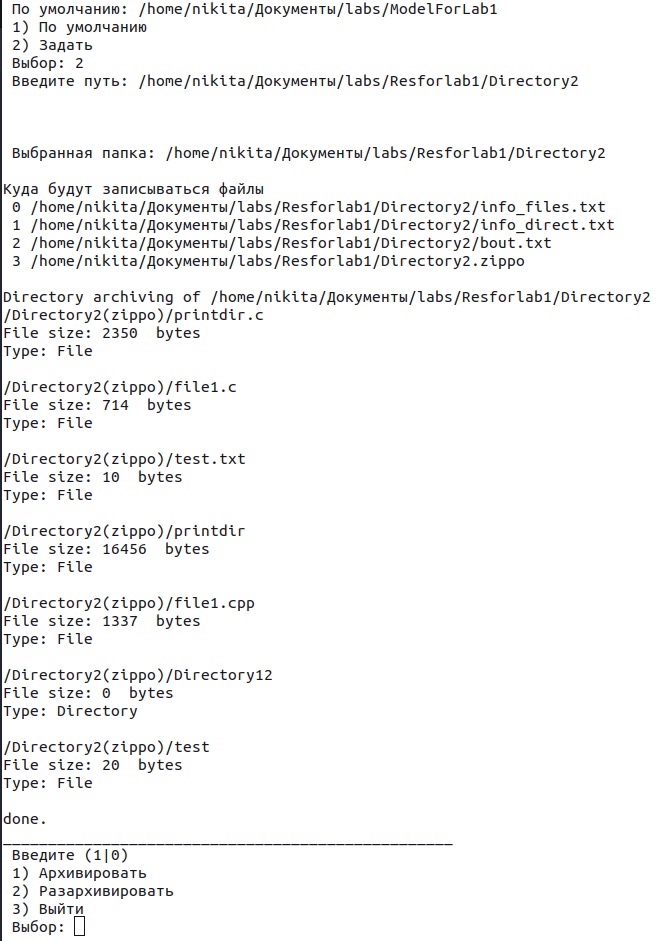


При выборе "Задать" предлагается написать директорию, если такой директории нет, то программа пишет о ошибке и просит ввести директорию ещё раз

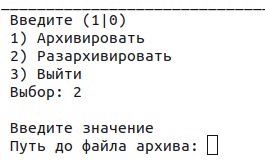


При вводе корректной директории отображаются файлы и сопутствующая информация, программа выходит в главное меню и создаётся архив

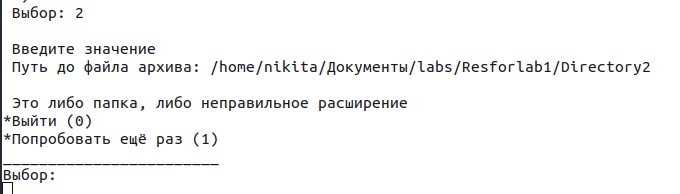




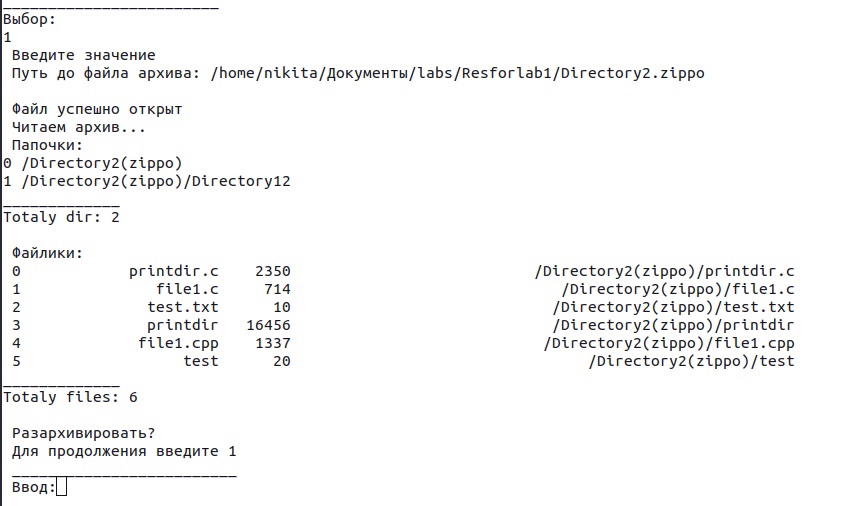
При выборе пункта "Разархивировать", программа просит ввести полный путь до файла архива



При неверном указании директории до файла программа выводит сообщение об ошибке на экран

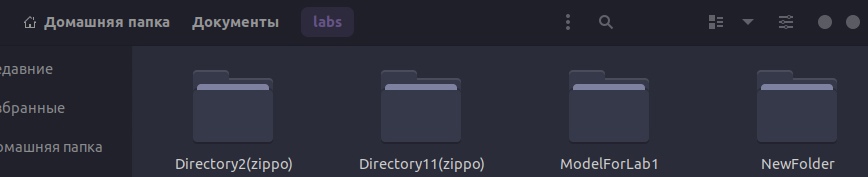


При верном указании директории отображается информация о файлах, содержащихся в архиве и появляется для разархивации



При выборе разархивации программа просит указать директорию, куда разархивировать данные





Выбор 3 в главном меню завершает работу программы

