**МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Отчет по лабораторной работе №1

по учебной дисциплине

«Операционные системы»

Выполнил студент группы М3О-310Б-20  
Рудаков Н.В.

Приняли Титов Ю.П., Грабовский М.Н.

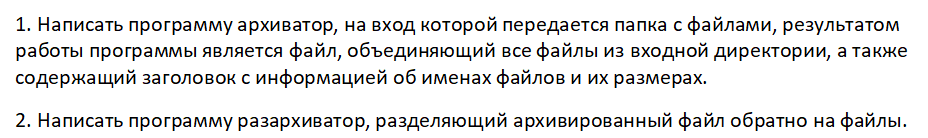
Оглавление

[Задание 3](#_Toc24064)

[Код программы 5](#_Toc29964)

[Результат работы программы 10](#_Toc31682)

## Задание



**Описание алгоритма**

Для того, чтобы совместить данные в файле в один файл и восстановить структуру(название файлов, папки в которых они расположены) создается дополнительный файл info.txt, который не является частью архиватора, в конце он удаляется.  
Данный файл содержит в себе название файлов и их размер.  
Безымянный  
  
Сначала идет общий размер информационной части, далее идет название папки, которую пользователь захотел архивировать  
После этого идут два флага - первое число это уровень иерархии, на котором мы находимся, второе число это флаг: 1 - папка, 0 - файл  
Если объект представляет собой папку - стоит 1 - далее идет название папки и снова флаги  
Если объект - файл, то следом идет размер файла и название файла и снова флаги.  
Данный информационный файл будет записываться в начало файла-архива, который составляется, а уже потом данные всех файлов.  
Таким образом нужно выполнить 2 раза обход всей директории, который указал пользователь.  
Первый обход - составление информационного файла.  
Второй обход - добавление в файл-архив данных из файлов, который нашли при обходе.  
  
Восстановление иерархичной структуры делается следующим образом:  
Сначала читается общий размер информационной части  
Далее читается название главной папки, создается папка с таким названием и делается переход в нее с помощью функции chdir()  
Далее считывает файл-архив пока не достигнем конца информационной части  
Если перед нами папка, то создаем ее, запоминаем уровень и название в трехмерный массив\*, который будет отслеживать связь уровня и название папки  
Если перед нами файл, то с помощью fteel запоминаем текущий файловый указатель и с fseek прыгаем на то место, где начинается файл, и читаем ровно столько данных, сколько было указано перед названием папки.  
Затем снова с помощью fteel возвращаем файловый указатель.  
  
  
\*Трехмерный символьный массив нужен для того, чтобы запоминать какая была последняя папка, на каком уровне иерархии  
Допустим мы создали папки и находимся на 3 уровне иерархии, а следующий файл в информацинной части говорит, что он находится на 0 уровне иерархии. С помощью данного трехмерного массива можно легко переместиться как назад, так и вперед по иерархии.

# **Код программы**

**Файл packer.h**

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

/\*Функция получения размера файла\*/  
off\_t fsize(const char \*filename)  
{  
 off\_t \_file\_size;

struct stat \_fileStatbuff;  
 int fd = open(filename, O\_RDONLY);  
 if(fd == -1)  
 {  
 printf("Ошибка при открытии файла %s\n", filename);  
 return -1;  
 }  
 if ((fstat(fd, &\_fileStatbuff) != 0) || (!S\_ISREG(\_fileStatbuff.st\_mode)))   
 {  
 printf("Ошибка при получении информации о файле %s\n", filename);  
 return -1;  
 }  
 else  
 {  
 \_file\_size = \_fileStatbuff.st\_size;  
 }  
 close(fd);  
 return \_file\_size;  
}

/\*Процедура записи в начало информационной части название папки\*/  
void dir\_info(char\* path)  
{  
 FILE\* info = fopen("info.txt","a");  
 char separator = '|';  
 fputc(separator, info);  
 fprintf(info,"%s", path);  
 fputc(separator, info);  
 fclose(info);  
}

/\*Процедура записи размера и названия файла в информационную часть\*/  
void file\_info(char\* pwd,char\* path, char\* filename, int depth, int dir)  
{  
 char pwd\_file\_info[1024];  
 strcpy(pwd\_file\_info, pwd);  
 strcat(pwd\_file\_info, "/info.txt");  
 struct stat statbuf;  
 off\_t size;  
 lstat(path, &statbuf);  
 FILE\* info = fopen(pwd\_file\_info,"a");  
 char separator = '|';  
 if(info == NULL && !S\_ISDIR(statbuf.st\_mode))  
 {

printf("Ошибка при открытии файла\n");  
 return;  
 }  
 fprintf(info, "%ld", depth); //Уровень иерархии  
 fputc(separator,info);  
 fprintf(info, "%ld", dir); //Флаг папки или файла  
 fputc(separator,info);  
 if(dir == 1) //Папка - название папки  
 {  
 fprintf(info,"%s",filename);  
 fputc(separator,info);  
 }  
 if(dir == 0) //Файл - размер файла и название  
 {  
 if((size = fsize(path)) == -1)  
 return;  
 fprintf(info, "%ld", size);  
 fputc(separator,info);  
 fprintf(info, "%s", filename);  
 fputc(separator,info);  
 }  
 fclose(info);  
}

/\*Функция обхода папки и записи в информационную часть необходимые данные\*/  
int get\_file\_info(char\* pwd,char\* argv,int depth, int flag)  
{  
 struct dirent \*pDirent;  
 DIR \*pDir;  
 struct stat statbuf;  
 pDir = opendir(argv);  
 if(pDir == NULL)  
 {  
 printf("Cannot open directory '%s'\n", argv);  
 return 1;  
 }  
 //Получить название папки  
 if(flag == 1)  
 dir\_info(argv);  
 chdir(argv);

while ((pDirent = readdir(pDir)) != NULL)   
 {  
 lstat(pDirent->d\_name, &statbuf);  
 if (strcmp(pDirent->d\_name, ".") == 0 || strcmp(pDirent->d\_name, "..") == 0) continue;  
 char BufStr[1024];  
 getcwd(BufStr,sizeof(BufStr));

if(strcmp(BufStr, pwd) == 0)  
 continue;

if(S\_ISDIR(statbuf.st\_mode))  
 {  
 file\_info(pwd,BufStr,pDirent->d\_name,depth,1);  
 get\_file\_info(pwd,pDirent->d\_name, depth + 1,0);  
 }  
 else  
 {  
 strcat(BufStr, "/");  
 strcat(BufStr, pDirent->d\_name);  
 printf("[\*] file: \t%s\n", BufStr);

file\_info(pwd,BufStr,pDirent->d\_name,depth,0);  
 }

}  
 chdir("..");  
 closedir (pDir);  
 return 0;

}

/\*Процедура создания файла\*/  
void create\_file\_paker(char\* pwd\_file\_packer,char\* argv,FILE\* f, FILE\* final)  
{  
 struct dirent \*pDirent;  
 DIR \*pDir;  
 struct stat statbuf;  
 pDir = opendir(argv);  
 char ch;  
 chdir(argv);

while ((pDirent = readdir(pDir)) != NULL)   
 {  
 lstat(pDirent->d\_name, &statbuf);  
 if (strcmp(pDirent->d\_name, ".") == 0 || strcmp(pDirent->d\_name, "..") == 0) continue;  
 char BufStr[1024];  
 getcwd(BufStr,sizeof(BufStr));  
 if(S\_ISDIR(statbuf.st\_mode))  
 create\_file\_paker(pwd\_file\_packer, pDirent->d\_name,f,final);  
 else  
 {  
 strcat(BufStr, "/");  
 strcat(BufStr, pDirent->d\_name);  
 f = fopen(BufStr, "rb");  
 final = fopen(pwd\_file\_packer, "ab");  
 while((ch = getc(f))!= EOF)  
 {  
 putc(ch,final);  
 }  
 fclose(final);  
 fclose(f);  
 }

}  
chdir("..");  
closedir(pDir);

}

/\*Функция создания файла-архива\*/  
int packer(char\* pwd, char\* argv,int flag)  
{  
 char pwd\_file\_packer[1024], pwd\_file\_info[1024];  
 strcpy(pwd\_file\_packer, pwd);  
 strcpy(pwd\_file\_info, pwd);  
 strcat(pwd\_file\_info, "/info.txt");  
 strcat(pwd\_file\_packer, "/file.paker");  
 get\_file\_info(pwd,argv, 0, 1);  
 char ch;

FILE\* info = fopen(pwd\_file\_info,"rb");  
 FILE\* f;  
 FILE\* final = fopen(pwd\_file\_packer, "ab");

if(info == NULL || final == NULL)  
 {  
 printf("Ошибка при открытии/создании файла\n");  
 return 1;  
 }  
 int info\_fsize = fsize(pwd\_file\_info);  
 fprintf(final, "%ld", fsize(pwd\_file\_info));  
 char\* buf\_info = (char\*)malloc(info\_fsize \* sizeof(char));  
 if(buf\_info == NULL)  
 {  
 printf("Не удалось выделить память\n");  
 return 1;  
 }  
 fread(buf\_info, info\_fsize \* sizeof(char),1,info);  
 fwrite(buf\_info, info\_fsize \* sizeof(char),1,final);

if(remove("info.txt") == -1)  
 {  
 printf("Не удалось удалить info.txt");  
 return 1;  
 }  
 fclose(info);  
 fclose(final);  
 free(buf\_info);  
 create\_file\_paker(pwd\_file\_packer,argv,f,final);

return 0;  
}

**Файл unpacker.h**

#include <regex.h>

/\*Функция создания папки  
\*Возвращает длину названия папки  
\*В случае неудачи - возвращает 0\*/  
off\_t create\_dir(FILE\* f,char\* argv)  
{  
 char ch; //символ, который считываем  
 char\* head\_argv = argv; //указывает на начало названия папки

int len\_of\_name\_dir = 0; //размер названия папки

/\*считываем посимвольно название папки\*/  
 while((ch = getc(f)) != '|')  
 {  
 \*argv++ = ch;  
 len\_of\_name\_dir++;  
 }  
 \*argv = '\0';  
 if(mkdir(head\_argv, 0777) == -1)  
 {  
 printf("Не получилось создать папку %s\n", head\_argv);  
 return 0;  
 }  
 return len\_of\_name\_dir;  
}

/\*Функция создания файла из архива  
\*Возвращает размер названия файла  
\*В случае неудачи возвращает 1\*/  
off\_t create\_file(FILE\* f, off\_t positionOfFile,off\_t sizeOfInfo,off\_t sizeOfFile, char\* argv)  
{  
 char ch; //Символ, который считываем  
 char\* head\_argv = argv; //Начало название файла  
 off\_t result = 0; //Длина названия файла

/\*Считываем название файла\*/  
 while((ch = getc(f)) != '|')  
 {  
 \*argv++ = ch;  
 result++;  
 }  
 \*argv = '\0';

FILE\* cf = fopen(head\_argv, "w");  
 if(cf == NULL)  
 {  
 printf("Не удалось создать файл %s\n", head\_argv);  
 return 1;  
 }  
 off\_t pos = ftell(f); //Указатель на текущую позицию

/\*Прыгаем на начало адресной части файла, который должен быть разахивирован\*/  
 if(fseek(f, positionOfFile + sizeOfInfo,SEEK\_SET) != 0)  
 {  
 if(errno == EINVAL)  
 {  
 printf("Неправильно указана позиция начала смещения fseek\n");  
 }  
 if(errno == ESPIPE)  
 {  
 printf("Неправильно указано смещение fseek\n");  
 }  
 return 1;  
}

/\*Считываем с начала адресной части файла-архива в файл, который создали - восстанавливаем данные\*/  
 char block[sizeOfFile];  
 fread(block,sizeof(block), 1,f);  
 fwrite(block,sizeof(block),1,cf);  
 fclose(cf);  
 /\*Возвращаем файловый указатель на информационную часть\*/  
 fseek(f,pos,SEEK\_SET);  
 return result;  
}

/\*Функция получения из информационной части размер файла в формате integer\*/  
off\_t ret\_size\_of\_file(FILE\* f, int\* cnt\_digs)  
{  
 char ch; //считываемый символ  
 off\_t result = 0; //само число, которое было получено из файла(328 из файла превратиться в 328 int)  
 int i = 0; //получаем ascii код цифры  
 while((ch = getc(f)) != '|')  
 {  
 int i = ch - 48;  
 result = result \* 10 + i;  
 if(cnt\_digs != NULL)  
 \*cnt\_digs = \*cnt\_digs + 1;//считаем количество цифр  
 }  
 return result;  
}

/\*Функция для проверки правильного формата подачи файла-архива\*/  
int regex\_file(char\* argv)  
{  
 regex\_t regex;  
 int reti;  
 char msgbuf[100];

/\*все файлы, которые имеют расширения .paker\*/  
 reti = regcomp(&regex, "[[:alnum:]]+\\.paker", REG\_EXTENDED);  
 if (reti)  
 {  
 printf("Не удалось составить регулярное выражение\n");  
 return 1;  
 }

reti = regexec(&regex, argv, 0, NULL, 0);  
 if (!reti)   
 {  
 regfree (&regex);  
 return 0;  
 }  
 else if (reti == REG\_NOMATCH)  
 {  
 printf("Файл не соотвествует формату \"name-file\".paker");  
 regfree (&regex);  
 return 1;  
 }  
 else   
 {

regerror(reti, &regex, msgbuf, sizeof(msgbuf));  
 printf("Regex match failed: %s\n", msgbuf);  
 return 1;  
 }

}

/\*Функция распаковки\*/  
int unpacker(char\* argv)  
{

/\*Аргумент командной строки не соотвествует файлу-архиву\*/  
 if(regex\_file(argv) == 1)  
 return 1;

FILE\* main = fopen(argv, "rb");  
 if(main == NULL)  
 {  
 printf("Не удалось открыть файл %s", argv);  
 return 1;  
 }

int cnt\_digs = 0;  
 int stop\_cnt\_digs = 0;  
 int level = 0;  
 off\_t sizeOfInfo = ret\_size\_of\_file(main, &cnt\_digs);  
 char nameOfDir[1024];  
 off\_t len\_of\_name\_dir = create\_dir(main, nameOfDir);  
 chdir(nameOfDir);  
 int i = cnt\_digs + len\_of\_name\_dir + 2;

char Folders[5][5][1024];  
 int cntElementsEachLevel[5];  
 memset(cntElementsEachLevel, 0, sizeof(cntElementsEachLevel));  
 int sizeOfPositionOfFile = 0;  
 int cnt\_level = 0;  
 while(i < sizeOfInfo)  
 {

off\_t read\_level = ret\_size\_of\_file(main, NULL);  
 i += 2;  
 off\_t read\_flags = ret\_size\_of\_file(main, NULL);  
 i += 2;

if(read\_level > level)  
 {

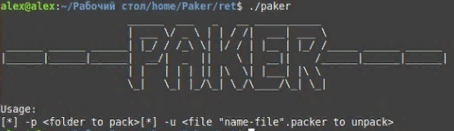
chdir(Folders[level][cntElementsEachLevel[level]]);  
 level = read\_level;  
 }

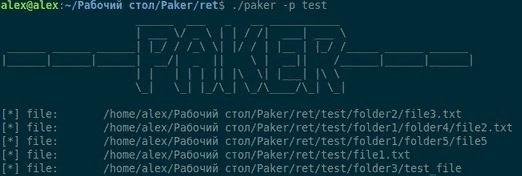
while(read\_level < level)  
 {

chdir("..");  
 level--;  
 }

if(read\_level == level)  
 cntElementsEachLevel[level] += 1;  
 if(read\_flags == 1)  
 {  
 char nameOfFolder[1024];  
 off\_t lenOfNameFolder = create\_dir(main, nameOfFolder);  
 i += lenOfNameFolder + 1;  
 strcpy(Folders[level][cntElementsEachLevel[level]], nameOfFolder);  
 }  
 if(read\_flags == 0)  
 {  
 char nameOfFile[1024];  
 off\_t sizeOfFile = ret\_size\_of\_file(main, &stop\_cnt\_digs);  
 off\_t lenOfFileSize = create\_file(main, cnt\_digs + sizeOfPositionOfFile,sizeOfInfo,sizeOfFile,nameOfFile);  
 sizeOfPositionOfFile += sizeOfFile;  
 i += lenOfFileSize + stop\_cnt\_digs;  
 }  
 }  
 return 0;  
}

# Результат работы программы

Если запустить программу без параметров, то выведется help информация  
  
  
Для архивирования папки нужно, чтобы программа была в той же директории, что и папка, и указать только название папки

  
При разархировании необходимо, чтобы папки с названием, которая является корневой в архиве, не было в директории, откуда запускается программа  
  
  
Если пользователь указал все верно, то в директории, где находится файл-архив и программа появится папка с разархивированными данными  
  
