Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа № 15

Архиватор файлов

Выполнил студент группы № M3109
Бабич Артём Антонович
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург
2020

Текст задания

Целью лабораторной работы является разработка программы по архивированию и распаковке нескольких файлов в один архив. Архиватор должен

1. Уметь архивировать несколько (один и более) указанных файлов в архив с расширением ***.arc**
2. Уметь распаковывать файловых архив, извлекая изначально запаковонные файлы
3. Предоставлять список файлов упакованных в архиве
4. *Сжимать и разжимать данные при архивировании с помощью алгоритма Хаффмана (опциональное задание, оценивается доп баллами)*

Архиватор должен быть выполнен в виде консольного приложения, принимающего в качестве аргументов следующий параметры

- **--file FILE**
  Имя файлового архива с которым будет работать архиватор
- **--create**
  Команда для создания файлового архива
- **--extract;**
  Команда для извлечения из файлового архива файлов
- **--list**
  Команда для предоставления списка файлов, хранящихся в архиве
- **FILE1 FILE2 …. FILEN**
  Свободные аргументы для передачи списка файлов для запаковки

Примеры использования:

arc --file data.arc --create a.txt b.bin c.bmp
arc --file data.arc --extract
arc --file data.arc --list

Решение с комментариями

Файл main.c (содержит главную функцию и подключенный заголовочник)

```c
#include <stdio.h>
#include <locale.h>

#include "arch.h"

int main(int argc, char *argv[]) {
    setlocale(LC_ALL, "Rus");

    char *arch_name = argv[2];

    switch (argv[3][2]) {
        case 'c':
            create(arch_name, argc, argv);
            break;
        case 'e':
            extract(arch_name);
            remove(arch_name);
            break;
        case 'l':
            printf("Файлы, хранящиеся в архиве %s:\n", arch_name);
            list(arch_name);
            break;
        default:
            puts("Была получена некорректная команда!");
            break;
    }


    return 0;
}
```

Файл arch.h (заголовочник, содержит юнионы хедеров архива и файла, а также прототипы функций)

```c
#ifndef LAB15_ARCH_H
#define LAB15_ARCH_H

union archive_header {
    char buffer[8];

    struct {
        char archive_type[4];
        unsigned int archive_size;
    } arch_data;
};

union file_header {
    char buffer[8];

    struct {
        unsigned int header_size;
        unsigned int file_size;
    } file_data;
};

void copy_file(FILE*, FILE*);

unsigned int get_file_size(char*);

void create(char*, int, char *[]);
```

```
void list(char*);

void extract(char*);

#endif //LAB15_ARCH_H
```

Файл arch.c (содержит функции для архивации, вывода списка файлов и распаковки архива)

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

#include "arch.h"

void copy_file(FILE *from, FILE *to) {
    int c;
    while ((c = getc(from)) != EOF) {
        putc(c, to);
    }
}

unsigned int get_file_size(char *file_name) {
    FILE *file = fopen(file_name, "rb");
    fseek(file, 0, SEEK_END);
    unsigned int file_size = ftell(file);
    fclose(file);
    return file_size;
}

void create(char *arch_name, int argc, char *argv[]) {
    FILE *arch_file = fopen(arch_name, "wb");

    union archive_header arc_header;
    arc_header.arch_data.archive_type[0] = 'F';
    arc_header.arch_data.archive_type[1] = 'A';
    arc_header.arch_data.archive_type[2] = 'R';
    arc_header.arch_data.archive_type[3] = 'C';
    arc_header.arch_data.archive_size = 8;
    for (int i = 4; i < argc; i++) {
        arc_header.arch_data.archive_size += get_file_size(argv[i]) + 8 +
strlen(argv[i]);
    }
    fwrite(arc_header.buffer, sizeof(char), 8, arch_file);

    for (int i = 4; i < argc; i++) {
        union file_header file_header;

        file_header.file_data.file_size = get_file_size(argv[i]);
        file_header.file_data.header_size = 8 + strlen(argv[i]);
        fwrite(file_header.buffer, sizeof(char), 8, arch_file);
        fwrite(argv[i], sizeof(char), strlen(argv[i]), arch_file);

        FILE *file;
        file = fopen(argv[i], "rb");

        copy_file(file, arch_file);

        fclose(file);
    }
    fclose(arch_file);
}
```

```
void list(char *arch_name) {
    FILE *arch_file = fopen(arch_name, "rb");

    union archive_header arc_header;
    fread(arc_header.buffer, sizeof(char), 8, arch_file);

    while (ftell(arch_file) < arc_header.arch_data.archive_size) {
        union file_header file_header;
        fread(file_header.buffer, sizeof(char), 8, arch_file);

        char *file_name = malloc((file_header.file_data.header_size - 8) *
sizeof(char));
        fread(file_name, sizeof(char), file_header.file_data.header_size - 8,
arch_file);

        printf("%s\n", file_name);
        fseek(arch_file, file_header.file_data.file_size, SEEK_CUR);
        free(file_name);
    }

    fclose(arch_file);
}

void extract(char *arch_name) {
    FILE *arch_file = fopen(arch_name, "rb");

    union archive_header arc_header;
    fread(arc_header.buffer, sizeof(char), 8, arch_file);

    while (ftell(arch_file) < arc_header.arch_data.archive_size) {
        union file_header file_header;
        fread(file_header.buffer, sizeof(char), 8, arch_file);

        char *file_name = malloc((file_header.file_data.header_size - 8) *
sizeof(char));
        fread(file_name, sizeof(char), file_header.file_data.header_size - 8,
arch_file);

        FILE *output_file = fopen(file_name, "wb");

        for (unsigned int i = 0; i < file_header.file_data.file_size; i++) {
            int c = getc(arch_file);
            putc(c, output_file);
        }

        fclose(output_file);
        free(file_name);
    }
    fclose(arch_file);
}
```

Данный код содержит в себе реализацию простого архиватора без методов сжатия данных, выполняет три функции:

- Архивация (create)
- Список файлов (list)
- Распаковка (extract)