Министр науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа № 13

Редактор метаинформации mp3-файла

Выполнила студентка группы № M31XX
Черных Арсений Игоревич
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург
2023

Текст задания

Задача.

Реализовать редактор текстовой метаинформации mp3 файла. В качестве стандарта метаинформации принимаем ID3v2.

Редактор представлять из себя консольную программу принимающую в качестве аргументов имя файла через параметра --*filepath* , а также одну из выбранных команд

1. --show - отображение всей метаинформации в виде таблицы
2. *--set=prop_name --value=prop_value* - выставляет значение определенного поля метаинформации с именем prop_name в значение prop_value
3. --get=prop_name - вывести определенное поле метаинформации с именем prop_name

Например:

app.exe --filepath=Song.mp3 --show
app.exe --filepath=Song.mp3 --get=TIT2
app.exe --filepath=Song.mp3 --set=COMM --value=Test

Примечание.

При выполнения данной работы разрешается использовать только стандартную библиотеку языка С. Исключением может являться процесс разбора аргументов командной строки.

Решение с комментариями:
Программа определяет три структуры: Header (заголовок), ExtendedHeader (расширенный заголовок) и Frame (кадр) для представления соответственно заголовка MP3 файла, его расширенного заголовка и отдельных кадров. CalcNewFrameSize: Рассчитывает и устанавливает новый размер кадра на основе указанного размера значения.
PrintFrame: Выводит содержимое кадра из указанного файла.
CalcFrameSize: Рассчитывает размер кадра на основе входных данных. ShowAllFrames: Отображает информацию о всех кадрах в MP3 файле.
GetFrame: Выводит информацию о конкретном кадре по его имени.
SetValue: Устанавливает новое значение для указанного свойства в MP3 файле.
Программа предоставляет команды командной строки (--show, --get, --set) для отображения всех кадров, получения информации о конкретном кадре и установки нового значения для свойства. Команды командной строки используются для указания действий: --show для отображения всех кадров, --get для получения информации о конкретном кадре, и --set для установки нового значения для свойства. Код также содержит операции с файлами для чтения и записи, а также манипуляции с байтами для работы с размерами и содержимым кадров MP3 файла.

```
#include <stdio.h>
#include <string.h>
#include <math.h>
struct Header {
    char tag[3];
    char ver;
    char subver;
    char flag;
    char size[4];
};
struct ExtendedHeader {
    char size[4];
    char flags[2];
};
struct Frame {
    char name[4];
    char size[4];
    char flags[2];
};
//функция для подсчета нового размера фрейма, его установки
void CalcNewFrameSize(char* frame_size, int value_size) {
    for (int i = 0; i < 4; i++) {
        frame_size[i] = value_size / pow(2, 7 * (3 - i));
        value_size -= frame_size[i] * pow(2, 7 * (3 - i));
    }
}
// функция для вывода фрейма
void PrintFrame(FILE *f, long long int frame_size) {
    while (frame_size > 0) {
```

```
        char a = fgetc(f);
        if (a >= 32 && a <= 127)
            printf("%c", a);
        frame_size--;
    }
};
// функция для подсчета размера фрейма
int CalcFrameSize(char* arr) {
    int size = 0;
    unsigned char a;
    for (int i = 0; i < 4; i++) {
        a = arr[i];
        printf("%c",a);
        size += a * pow(2, 7 * (3 - i));
    }
    return size;
}
// проверка на наличие дополнительного заголовка
long long int CheckExtended(FILE *f, long long int counter, struct Header *main_header) {
    if (main_header->flag & 64) {
        struct ExtendedHeader Extended;
        fread(&Extended, 6, 1, f);
        long long int extended_size = CalcFrameSize(Extended.size);
        fseek(f, extended_size, SEEK_CUR);
        counter += 6 + extended_size;
    }
    return counter;
}

void ShowAllFrames(const char *filename) {
    FILE *f = fopen(filename, "rb");
    struct Header main_header;
    fread(&main_header, 10, 1, f);
    long long frames_size = CalcFrameSize(main_header.size);
    long long counter = 0;
    counter = CheckExtended(f, counter, &main_header);
    struct Frame frame;
    while (counter <= frames_size) {
        fread(&frame, sizeof(frame), 1, f);
        long long frame_size = CalcFrameSize(frame.size);
        counter += 10 + frame_size;
        if (strcmp(frame.name, "APIC") == 0) {
            fseek(f, frame_size, SEEK_CUR);
        }
        else if ((frame.name[0] >= 65) && (frame.name[0] <= 91) && (strcmp(frame.name, "APIC") != 0)) {
            printf("%s ", frame.name);
            PrintFrame(f, frame_size);
            printf("\n");
        }
        else {
            break;
        }
    }
    fclose(f);
}
void GetFrame(const char *filename, const char* prop_name) {
    FILE *f = fopen(filename, "rb");
    struct Header main_header;
    fread(&main_header, 10, 1, f);
    long long frames_size = CalcFrameSize(main_header.size);
    long long counter = 0;
    counter = CheckExtended(f, counter, &main_header);
    struct Frame frame;
    while (counter <= frames_size) {
        fread(&frame, sizeof(frame), 1, f);
        long long frame_size = CalcFrameSize(frame.size);
```

```
        counter += 10 + frame_size;
        if ((strcmp(frame.name, prop_name) == 0)) {
            printf("%s ", frame.name);
            PrintFrame(f, frame_size);
            printf("\n");
        } else {
            fseek(f, frame_size, SEEK_CUR);
        }
    }
    fclose(f);
}
void SetValue(const char *filename, const char* prop_name, char* value) {
    FILE *f = fopen(filename, "rb+");
    FILE *new_file = fopen("newfile.mp3", "wb+");
    struct Header main_header;
    fread(&main_header, 10, 1, f);
    long long int frames_size = CalcFrameSize(main_header.size);
    long long int counter = 0;
    counter = CheckExtended(f, counter, &main_header);
    struct Frame frame;
    while (counter <= frames_size) {
        int frame_pos = ftell(f);
        fread(&frame, sizeof(frame), 1, f);
        long long int frame_size = CalcFrameSize(frame.size);
        counter += 10 + frame_size;
        if ((strcmp(frame.name, prop_name) == 0)) {
            long long int old_frame_size = frame_size;
            CalcNewFrameSize(main_header.size, CalcFrameSize(main_header.size) - frame_size + strlen(value));
            int current = ftell(f);
            fseek(f, old_frame_size, SEEK_CUR);
            unsigned char a = fgetc(f);
            while (!feof(f)) {
                fwrite(&a, 1, 1, new_file);
                a = fgetc(f);
            }
            fseek(f, 6, SEEK_SET);
            fwrite(main_header.size, sizeof(main_header.size), 1, f);
            CalcNewFrameSize(frame.size, strlen(value));
            fseek(f, frame_pos, SEEK_SET);
            fwrite(&frame, sizeof(frame), 1, f);
            fseek(f, current, SEEK_SET);
            fwrite(value, strlen(value), 1, f);
            fseek(new_file, 0, SEEK_SET);
            unsigned char t = fgetc(new_file);
            while (!feof(new_file)) {
                fwrite(&t, 1, 1, f);
                t = fgetc(new_file);
            }
        }
        else if (frame_size >= 0) {
            fseek(f, frame_size, SEEK_CUR);
        }
    }
    fclose(f);
    fclose(new_file);
}


int main(int argc, char **argv){
    char* filename = NULL;
    char* prop_name = NULL;
    char* value = NULL;
    for (int i = 0; i < argc; i++) {
        filename = argv[1] + 11;
        if (strcmp("--show", argv[i]) == 0) {
            ShowAllFrames(filename);
```

```
        }
        if (strncmp("--get", argv[i], 5) == 0) {
            prop_name = argv[i] + 6;
            GetFrame(filename, prop_name);
        }
        if (strncmp("--set", argv[i], 5) == 0) {
            prop_name = argv[i] + 6;
            value = argv[i+1] + 8;
            SetValue(filename, prop_name, value);
        }
    }
    return 0;
}
```