信息隐藏技术第二次大作业 —— 媒体文件格式剖析

学号：2013921
姓名：周延霖
专业：信息安全

一、概念介绍

1.实验要求

1. 任选一种媒体文件，进行格式剖析（建议使用UltraEdit）
2. 针对该类型的文件，讨论可能的隐藏位置和隐藏方法
3. 实现秘密信息的隐藏和提取

> 在本次实验中本人选择对音频文件进行分析

2.音频文件格式总览

说到语音技术就不得不说起音频数据，从硬件设备采集语音信号，语音信号的处理，到语音信号A/D转换得到原始数据(`raw data`)，再到对原始数据进行编码得到音频文件，对音频文件解码进行播放。使用最多的几种音频格式有MP3、WMA、WAV、AAC、FLAC、APE、WV、ASF、VQF、MID、OGG、M4A、eAAC+

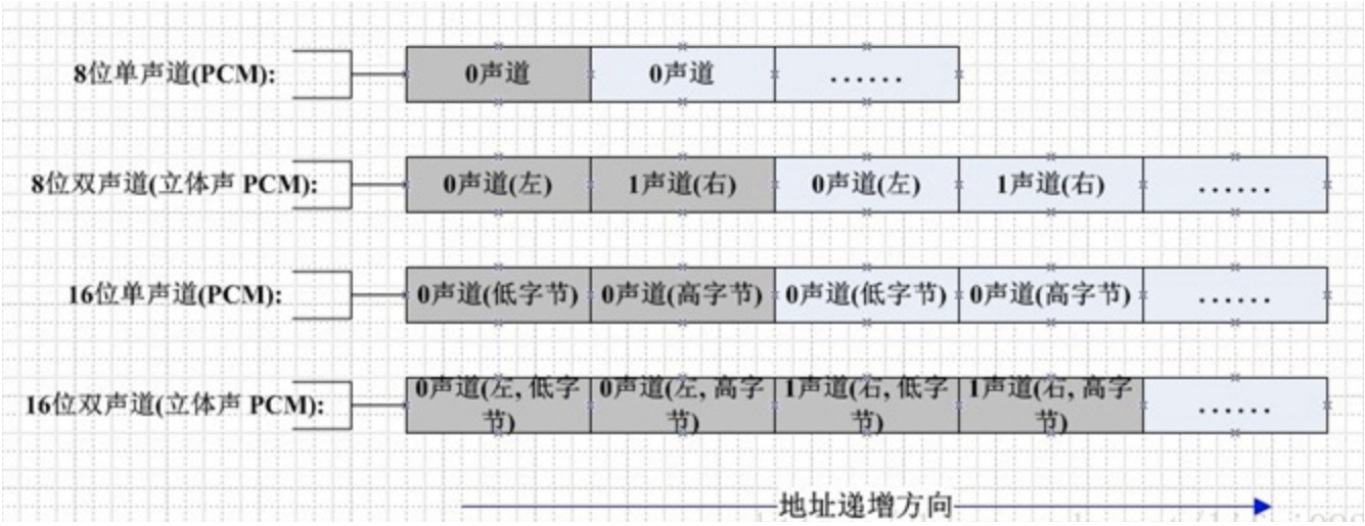
> 本次实验用到了`WAV`其中三种，故详细分析此格式文件

3.原始数据(`raw data`，以`PCM`编码为例)

模拟音频信号经模数转换（A/D变换）直接形成的二进制序列，该文件没有附加的文件头和文件结束标志

而PCM(Pulse-code Modulation，脉冲编码调制)是一种模拟信号的数字化方法，常被用于数字电信系统中，非常频繁地，PCM编码以一种串行通信的形式，使数字传讯由一点至下一点变得更容易————不论在已给定的系统内，或物理位置

PCM过程放在ALSA的实践中作为理论部分呈现，这里给出多通道音频数据的比特位特征：



如图所示，单通道音频数据以采样位数(bit)串行记录在比特流中:

1. 8 bit 采样位数: 意味着每个采样值能占据1个字节大小:

```
+------+------+------+------+------+------+------+------+------+
|  500 |  300 | -100 | -20  | -300 |  900 | -200 |  -50 |  250 |
+------+------+------+------+------+------+------+------+------+
```

2. 16 bit 采样位数：分为两个字节以小端(little-endian)方式存储在比特流中

双通道及多通道数据音频数据以采样位数组织如下(n 为采样位数)：

```
+-- n bit---+---n bit---+-----+---n bit---+---n bit--+------+--n bit---+--
-n bit---+----------+
|  channel1 |  channel2 | ... | channel1  | channel2 |  ... | channel1 |
channel2 |  ... |
+---------+---------+--------+---------+--------+---------+--------
+---------+---------+
```

PCM的每个样本值包含在一个整数i中，i的长度为容纳指定样本长度所需的最小字节数。首先存储低有效字节，表示样本幅度的位放在i的高有效位上，剩下的位置为0,计算机读入原始音频数据的方式与打开二进制文件相同，如下所示：

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

int main()
{
    char *fn;
    char *data;
    FILE *fp;
    int len;

    fn = "./test.pcm";
    fp = fopen(fn, "rb");
    if(!fp) {
        printf("file open failed.\n");
        exit(1);
    }
    fseek(fp, 0, SEEK_END);
    len = ftell(fp);
    if(!len) {
        printf("file is null.\n");
        fclose(fp);
        exit(1);
    }
    printf("file len = %d\n",len);
    data = (char *)malloc(len);
```

```
    fseek(fp, 0, SEEK_SET);
    fread(data, sizeof(__int16_t), len/sizeof(__int16_t), fp);
    fclose(fp);
    printf("%.*s\n",1,data);
    free(data);
    return 0;
}
```

4.wav文件

所有的电脑文件都是由0和1的二进制数字组成，音频文件自然也不例外，前面已经提到了音频文件是一个波形文件，并解析了音频文件如何从一个波形文件变成了0和1的二进制数字

WAV为微软公司（Microsoft)开发的一种声音文件格式，它符合RIFF(Resource Interchange File Format)文件规范，用于保存Windows平台的音频信息资源，被Windows平台及其应用程序所广泛支持，该格式也支持MSADPCM，CCITT A LAW等多种压缩运算法，支持多种音频数字，取样频率和声道，标准格式化的WAV文件和CD格式一样，也是44.1K的取样频率，16位量化数字，因此在声音文件质量和CD相差无几，WAV打开工具是WINDOWS的媒体播放器

wav文件的格式为：

```
struct waitor_wav_format
{
#pragma pack(1)
    char riff_id[4];                    //  "RIFF",big-endian
    __uint32_t file_len;                //  file length,little-endian
    char wave_id[4];                    //  "WAVE“,big-endian
    char fmt_id[4];                     //  "fmt",big-endian,beginning of
fmt chunk
    char transition[4];                 //  size of fmt chunk
    __uint16_t fmt_type;                //  1-PCM
    __uint16_t channel;                 //  通道数
    __uint16_t sample_rate;             //  采样率
    __uint32_t avg_bytes_per_sec;       //  sample_rate * block_align
    __uint16_t block_align;             //  每次采样大小
    __uint16_t bit_per_second;          //  采样精度
    //__uint16_t cbsize                   //  附加数据大小
    char data_id[4];                    //  "data"
    __uint32_t audio_len;               //  音频数据的长度
#pragma pack()
};
```

wav文件分为3个(或4个)chunk，每个chunk基本格式为 `chunk_id + chunk_size + chunk_data`。总体结构为 `文件头 wav_hdr + 音频数据` PCM格式。在解析wav文件时，可能会由于不同系统不同调制方式方式的不同增加几个额外的字节，这时应在数据结构中加入`#pragma pack(1)`，以设置对齐的方式空出额外长度，具体代码如下：

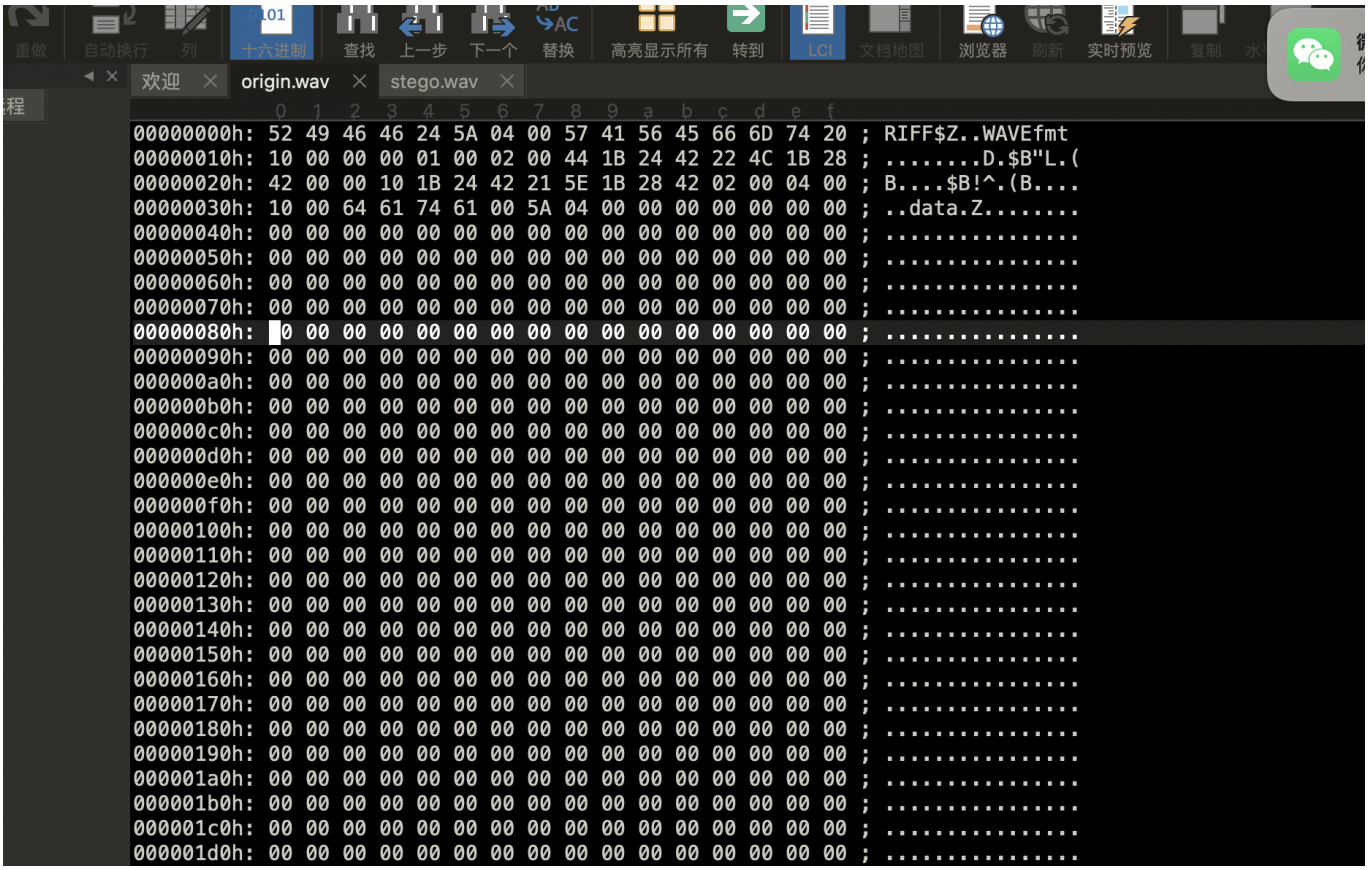
```
int main()
{
    waitor_wav_t *w;
    char *fn, *audio;
    // char buffer[4096];
    int ret;

    FILE *fp = fopen("./enc.ogg", "wb");

    fn = "../data/test.wav";
    w = waitor_wav_read(audio, fn);                    // 解析wav文件格式，转为二进制流voice
    if(!w) {
        printf("wavfile read failed.\n");
        fclose(fp);
        free(w);
        exit(1);
    }
    printf(" chunk_id1 : %.*s\n", 4, w->riff_id);
    printf(" file len : %d\n", w->file_len);
    printf(" fomat : %.*s\n", 4, w->wave_id);
    printf(" sub chunk id : %.*s\n", 4, w->fmt_id);
    printf(" fmt type : %d\n", w->fmt_type);
    printf(" channel : %d.\n", w->channel);
    printf(" sample_rate : %d \n", w->sample_rate);

    fclose(fp);
    free(w);
    return 0;
}
```

二、格式剖析

1.文件分析

首先打开文件简介，可以看到文件的一些信息如下所示：

然后用UltraEdit打开文件可以看到如下信息，将在后文对其进行分析



2.wav头文件详解

wav头文件的具体格式如下：

偏移地址	大小字节	数据块类型	内容
00H~03H	4	4字符	资源交换文件标志（RIFF）
04H~07H	4	长整数	从下个地址开始到文件尾的总字节数
08H~0BH	4	4字符	WAV文件标志（WAVE）
0CH~0FH	4	4字符	波形格式标志（fmt ），最后一位空格。
10H~13H	4	整数	过滤字节（一般为00000010H）
14H~15H	2	整数	格式种类（值为1时，表示数据为线性PCM编码）
16H~17H	2	整数	通道数，单声道为1，双声道为2
18H~1BH	4	长整数	采样频率
1CH~1FH	4	长整数	波形数据传输速率（每秒平均字节数）
20H~21H	2	整数	DATA数据块长度，字节。
22H~23H	2	整数	PCM位宽
24H~27H	4	4字符	“fact”,该部分一下是可选部分，即可能有，可能没有,一般到WAV文件由某些软件转换而成时，包含这部分。
28H~2BH	4	长整数	size,数值为4

偏移地址	字节数	类型	内容
24H~27H	4	4字符	数据标志符（data）
28H~2BH	4	长整型	DATA总数据长度字节
2CH...	...		DATA数据块

wav文件的三个最重要的参数是采样率，比特数，声道数

3.UltraEdit解析

下面对文件的具体信息进行详细分解：

如之前所列出来的origin.wav所示：

- 00H ~ 03H 52 49 46 46 对应的是RIFF
- 04H ~ 07H 0C 87 02 00 对应的是后面文件的大小
- 08H ~ 0BH 57 41 56 45 对应的是标示符WAVE
- 0CH ~ 0FH 66 6d 74 20 对应是波形格式标示符fmt
- 10H ~ 13H 10 00 00 00 对应的是过滤字节，不知道是什么作用，由于本文件不是标准的采样率，所以可能和上面的不一致
- 14H ~ 15H 01 00 对应的十进制是1 线性的PCM编码，我们这里只探讨PCM编码
- 16H ~ 17H 02 00 对应的十进制是2 表示双声道

- 18H ~ 1BH 44 1B 24 42 表示采样率
- 1CH ~ 1FH 22 4C 1B 28 波形数据传输率，每秒多少个字节
- 20H ~ 21H 42 00 为数据的调整数
- 22H ~ 23H 00 10 样本数据的位数

三、隐藏位置与方法

针对音频文件，本人想到了如下的隐藏思路：

- 频率颠簸隐藏：在音频文件中插入隐藏数据的一种方法是通过微调音频信号中的频率来实现。这种方法通常会在音频信号的高频段中插入隐藏信息，这样就可以避免影响听觉质量
- 时间颠簸隐藏：另一种常见的方法是通过微调音频信号中的时间来实现。这种方法通常会在音频信号的低频段中插入隐藏信息，这样就可以避免影响听觉质量
- 频率干扰隐藏：这种方法利用了音频信号的频率响应，通过在不影响听觉质量的情况下在音频信号的高频段中插入隐藏信息
- 时域干扰隐藏：这种方法通过在音频信号的短时幅度上微调隐藏信息，来在不影响听觉质量的情况下插入隐藏信息

需要注意的是，这些技术的效果取决于隐藏数据的大小、音频文件的质量以及其他一些因素。此外，这些技术也可能会被数字信号处理技术所检测到和防范，因此使用这些技术时需要慎重考虑

四、思路实现

以下是用MATLAB实现将信息隐藏在音频文件中，以及从中提取信息的代码：

1.隐藏信息

```
% 读取音频文件
[y, fs] = audioread('origin.wav');

% 将信息转换为二进制形式 2013921是我的学号
msg = '2013921';
msg_bin = dec2bin(msg, 8);

% 将二进制信息嵌入音频文件中
bit_idx = 1;
for i = 1:length(y)
    if bit_idx > length(msg_bin(:))
        break;
    end

    % 将音频文件样本转换为二进制形式
    y_bin = dec2bin(round(y(i)*32767), 16);

    % 将信息嵌入音频文件样本中
    y_bin(end) = msg_bin(bit_idx);
    bit_idx = bit_idx + 1;
```

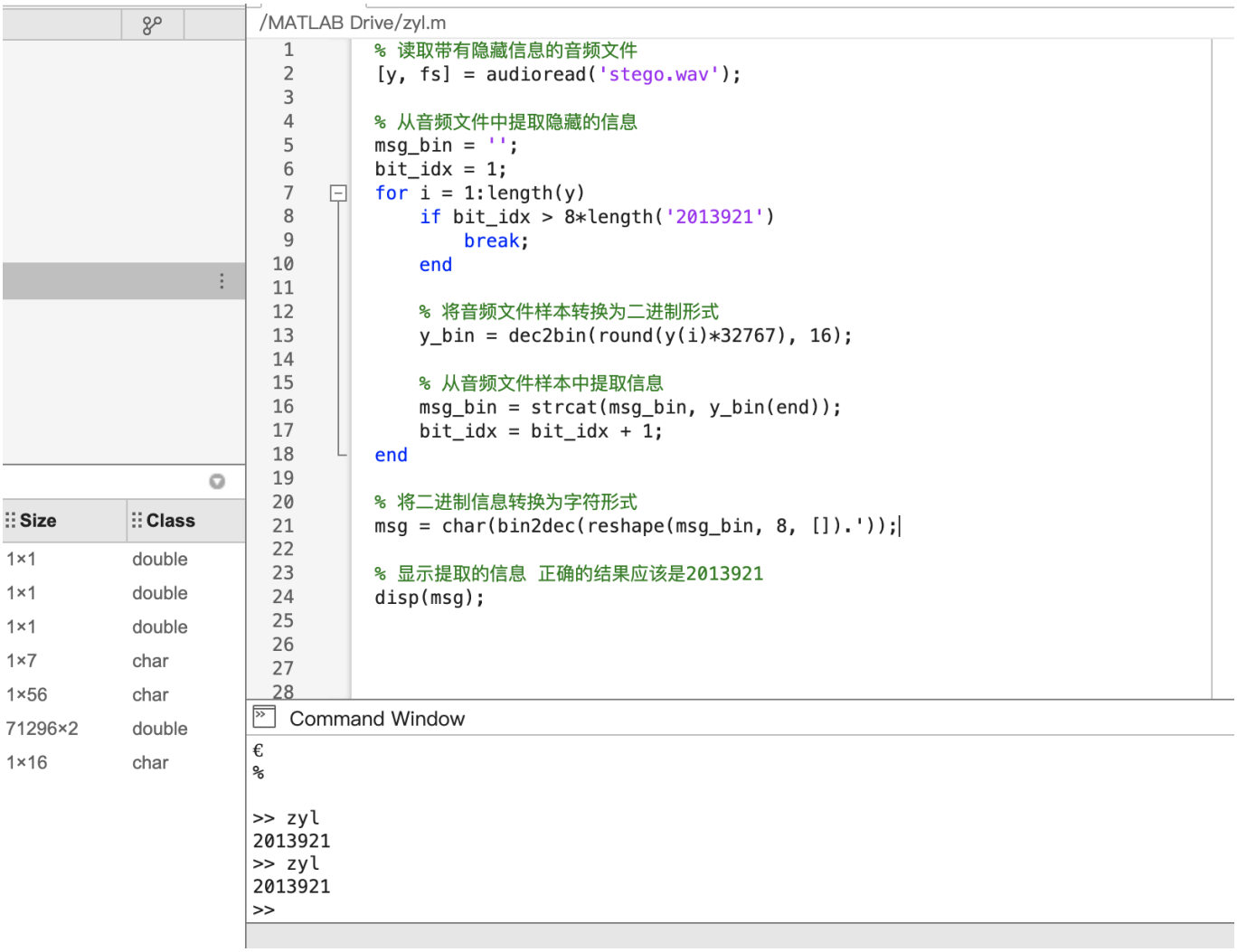
```
    % 将修改后的二进制样本转换回音频文件样本
    y(i) = bin2dec(y_bin)/32767;
end

% 将带有隐藏信息的音频文件保存
audiowrite('stego.wav', y, fs);
```

2.提取信息

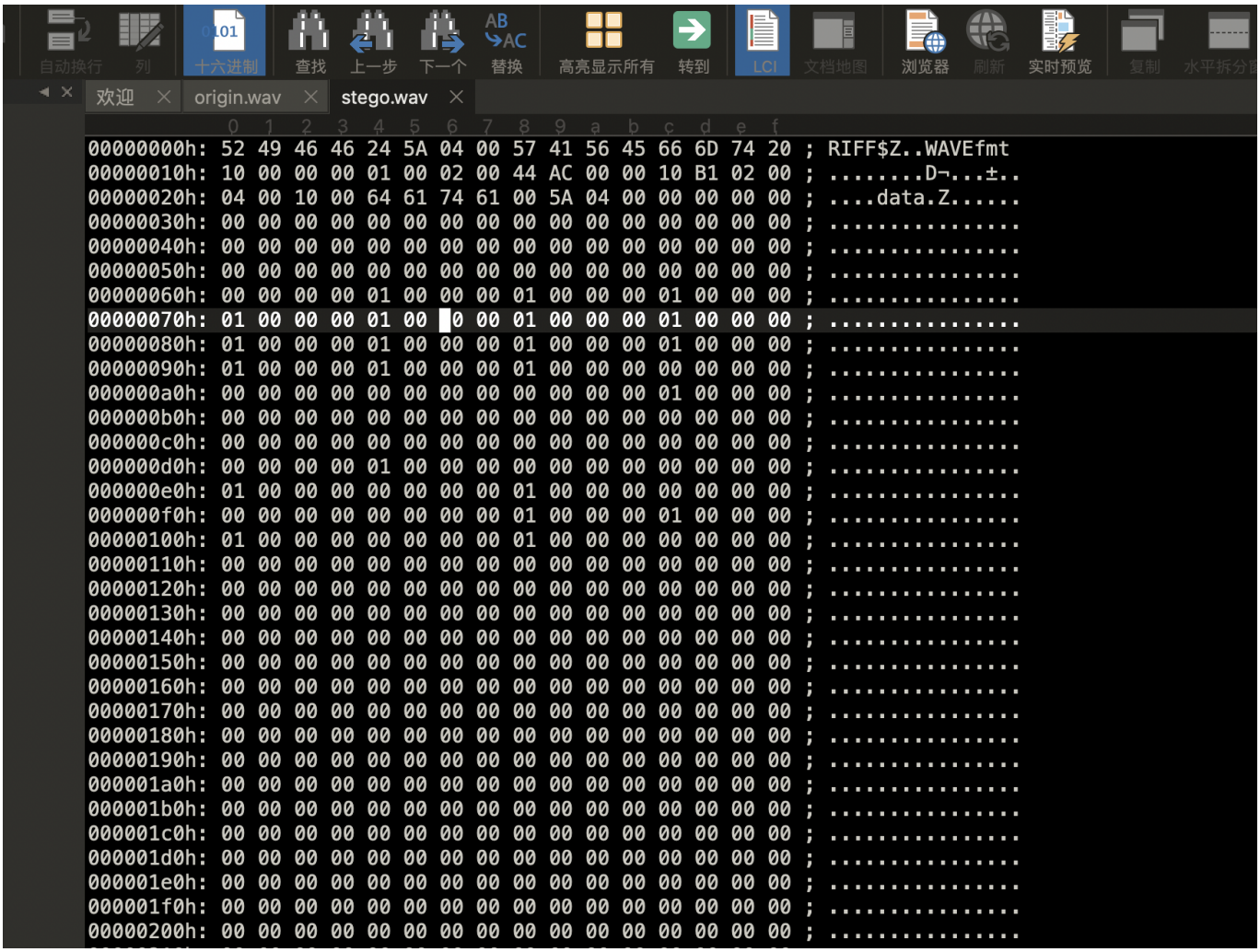
```
% 读取带有隐藏信息的音频文件
[y, fs] = audioread('stego.wav');

% 从音频文件中提取隐藏的信息
msg_bin = '';
bit_idx = 1;
for i = 1:length(y)
    if bit_idx > 8*length('2013921')
        break;
    end

    % 将音频文件样本转换为二进制形式
    y_bin = dec2bin(round(y(i)*32767), 16);

    % 从音频文件样本中提取信息
    msg_bin = strcat(msg_bin, y_bin(end));
    bit_idx = bit_idx + 1;
end

% 将二进制信息转换为字符形式
msg = char(bin2dec(reshape(msg_bin, 8, []).'));

% 显示提取的信息  正确的结果应该是2013921
disp(msg);
```

3.实验结果

在MATLAB中可以看到能成功提取信息2013921，如下图所示:



用UltraEdit打开插入隐藏信息的文件`stego.wav`，如下图所示：



分析如下：

- 00H ~ 03H 52 49 46 46 对应的是RIFF
- 04H ~ 07H 24 5A 04 00 对应的是后面文件的大小
- 08H ~ 0BH 57 41 56 45 对应的是标示符WAVE
- 0CH ~ 0FH 66 6d 74 20 对应是波形格式标示符fmt
- 10H ~ 13H 10 00 00 00 对应的是过滤字节，不知道是什么作用，由于本文件不是标准的采样率，所以可能和上面的不一致
- 14H ~ 15H 01 00 对应的十进制是1 线性的PCM编码，我们这里只探讨PCM编码
- 16H ~ 17H 02 00 对应的十进制是2 表示双声道
- 18H ~ 1BH 44 AC 00 00 表示采样率
- 1CH ~ 1FH 10 B1 02 00 波形数据传输率，每秒多少个字节
- 20H ~ 21H 04 00 为数据的调整数
- 22H ~ 23H 10 00 样本数据的位数

五、总结与展望

本次实验利用 MATLAB 将信息隐藏到音频文件中，并成功实现将其提取出来，但可能只适用于少量信息，太多的信息可能会被发现，将在未来对其进行优化，并且此方法仅适用于较小的信息，因为要在音频样本中嵌入足够的信息，可能会影响音频文件的质量

通过对所学到的理论知识进行相应的应用，对MATLAB的应用也更加的熟练，最后期待自己未来更好的发展，心想事成、万事胜意、未来可期