|  |  |  |
| --- | --- | --- |
| **ipn** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO** |  |

**Teoría de Comunicaciones y Señales**

**“Bajar Volumen a un Archivo WAV”**

Abstracto

El abstracto es un resumen preciso de todo el reporte completo. Su función es indicar los contenidos del reporte para que el lector pueda ver si vale la pena leerlo completo o no.

**Por:**

**Joel Mauricio Romero Gamarra**

Profesor:

EDUARDO GUTIÉRREZ ALDANA

Agosto 2017

**Índice**

Contenido

[Introducción: 1](#_Toc476313175)

[Análisis Teórico: 1](#_Toc476313176)

[Software (librarias, paquetes, herramientas): 2](#_Toc476313177)

[Procedimiento: 2](#_Toc476313178)

[Resultados 2](#_Toc476313179)

[Discusión: 2](#_Toc476313180)

[Conclusiones: 2](#_Toc476313181)

[Referencias: 3](#_Toc476313182)

[Código 3](#_Toc476313183)

# Introducción:

Lo que se pretendía al inicio de esta práctica, es bajar el volumen de un archivo wav a la mitad, por lo tanto, había que familiarizarse primero con este tipo de archivos y su manejo en lenguaje C para poder modificar su configuración (específicamente, el volumen).

El manejo de ciclos y la estructura para almacenar los contenidos de la cabecera es esencial para lograr un funcionamiento correcto y una señal de salida deseada.

# Análisis Teórico:

Un formato de archivo WAV es un subconjunto de las especificaciones RIFF capaz de almacenar muchos tipos de datos, pero principalmente multimedia (como audio y/o video), está formado por distintos bloques o fragmentos y cada uno de estos está señalado con una etiqueta de 4 caracteres como se muestra a continuación:

Contiene ‘SubChunk2Size’ datos reales y sonido del archivo WAV.

Contiene el tamaño de los datos reales a partir de este fragmento.

Contiene ‘data’ indicando el inicio del pedazo que contiene los datos reales y el sonido actual.

Contiene bits por muestra. Por ejemplo 8, 16, etc.

Contiene (SampleRate \* NumChannels \* BitsPerSample) / 8

Contiene bytes por muestra incluyendo todos los canales. (NumChannels \* BitsPerSample) / 8

Contiene 1 para Mono, 2 para Estéreo, etc.

Contiene frecuencia de muestreo, muestras por segundo. Por ejemplo 8,000, 44,100, etc.

Contiene ‘RIFF’ indicando el formato del archivo.

Tamaño total de todo el archivo – 8 bytes (por este segmento y el anterior de 4 bytes c/u).

Contiene ‘WAVE’ indicando el sub-formato del archivo.

Contiene ‘fmt ’ describiendo el formato de muestra e indicando el inicio del primer pedazo.

Contiene la longitud del resto del pedazo fmt.

Contiene 1 para PCM (Modulación por Impulsos Codificados), distinto de 1 tiene compresión.

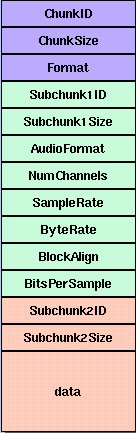


Figura 1. Cabecera de un archivo WAV 1

# Software (librarías, paquetes, herramientas):

Hacer una lista de TODOS los ítems utilizados en el laboratorio. Alternativamente, materiales pueden ser incluidos como parte del procedimiento.

# Procedimiento:

\* Diagramas de Flujo / Diagrama a Bloques  
\* Agregar detalles (paso a paso) del procedimiento de manera que cualquier persona que lea pueda repetir el experimento.

# Resultados

\* Esta sección debe incluir cualquier tabla de datos, observaciones, imágenes, etc.  
\* Todas las tablas y gráficas deben estar debidamente etiquetadas.

Esta sección describe, pero no explica los resultados obtenidos.

Puesto que se están presentando los resultados y no las cifras/figuras que representan los resultados, debemos asegurarnos de que nos referimos explícitamente a nuestros resultados y nada más, no sólo a las cifras/figuras (gráficas o tablas). Al describir resultados particulares en el texto de esta sección, **debemos asegurarnos de consultar la figura correspondiente entre paréntesis después de mencionarla en los resultados**. Las figuras deben ser insertadas en el texto lo más pronto posible después de haberlas mencionado.

# Discusión:

La sección de discusión tiene 2 objetivos principales:

* Interpretar y explicar los resultados del estudio.
* Explorar la importancia del estudio, encontrando, calificando y explorando la importancia teórica de los resultados.

La discusión es también un espacio en el reporte donde cualquier calificación o reservación que se tiene sobre la investigación debe ser mencionada,

# Conclusiones:

\*Lista una cosa que hayas aprendido y describe como lo aplicarías a una situación de la vida real.

\*Discute los posibles errores que podrían haber ocurrido en la colección de los datos (errores experimentales).

\*¿Cómo se aplicarían los resultados obtenidos generalmente?

\*¿Hubo algún defecto en el diseño experimental o en el procedimiento?

# Referencias:

**[1]** craig@ccrma.stanford.edu, ‘WAVE PCM sounfile format’. [Online]. Disponible en: <http://soundfile.sapp.org/doc/WaveFormat/>. [Accedido: 03 – septiembre – 2017].

# Código

**Cabecera.h**

**typedef** **struct** CABECERA

{

**char** ChunkID[4]; //Contiene las 'RIFF'

**unsigned** **int** ChunkSize; //Contiene el tamaño total sin contar este y el segmento anterior (8 bytes)

**unsigned** **char** Format[4]; //Contiene 'WAVE'

//Aqui comienza el primer subchunk 'fmt'

**char** SubChunk1ID[4]; //Contiene 'fmt'

**unsigned** **int** SubChunk1Size; //Contiene el tamaño del resto de el primer subchunk

**unsigned** **int** AudioFormat; //Formato de audio, es es distinto de 1, es forma de compresión

**unsigned** **int** NumChannels; //Numero de canales, mono = 1, estereo = 2, etc.

**unsigned** **int** SampleRate; //8000, 44100, etc.

**unsigned** **int** ByteRate; //(SampleRate \* Numero canales \* Bits per Sample) / 8

**unsigned** **int** BlockAlign; //(Numero canales \* Bits per Sample) / 8

**unsigned** **int** BitsPerSample; //8 bits, 16 bits, etc.

//Aqui comienza el segundo subchunk 'data'

**char** SubChunk2ID[4]; //Contiene 'data'

**unsigned** **int** SubChunk2Size; //Numero de bytes en los datos, es decir, bytes despues de este segmento

**unsigned** **int** data; //Datos de sonidos reales

}cabecera;

FILE \* abreArchivo (**char** \* nombreArch, **char** \* nombreModificado, **int** tipo); //Para abrir los archivos de entrada y salida

**void** leerCabecera (FILE \* archivoEntrada, FILE \* archivoSalida, cabecera \* cab); //Para copiar e imprimir la cabecera

**Cabecera.c**

#include <stdio.h>

#include <stdlib.h>

#include "Cabecera.h"

FILE \* abreArchivo (**char** \* nombreArch, **char** \* nombreModificado, **int** tipo)

{

FILE \* pt1, \* pt2;

pt1 = fopen (nombreArch,"rb");

**if** (pt1 == NULL)

{

printf("Error al abrir archivo '%s'.\n", nombreArch);

exit(0);

}

//Abrimos el archivo a escribir en modo binario

pt2 = fopen (nombreModificado,"wb");

**if** (pt2 == NULL)

{

printf("Error al crear el archivo '%s'.\n", nombreModificado);

exit(1);

}

**if** (tipo == 1)

{

printf("Archivo '%s' abierto correctamente.\n", nombreArch);

**return** pt1;

}

**else**

{

printf("Archivo '%s' creado correctamente.\n", nombreModificado);

**return** pt2;

}

}

**void** leerCabecera (FILE \* archivoEntrada, FILE \* archivoSalida, cabecera \* cab)

{

**int** lectura, escritura;

**unsigned** **char** buffer4[4], buffer2 [2], \* formatoArchivo;

formatoArchivo = (**char** \*) malloc (**sizeof** (**char**));

//ChunkID

lectura = fread(cab -> ChunkID,**sizeof**(cab -> ChunkID),1,archivoEntrada);

printf("\n\n\n(1-4) Chunk ID: %s\n\n",cab -> ChunkID);

escritura = fwrite (cab -> ChunkID, **sizeof** (cab -> ChunkID), 1, archivoSalida);

//ChunkSize

//Se convierten los datos a entero

lectura = fread(buffer4, **sizeof**(buffer4),1,archivoEntrada);

escritura = fwrite(buffer4, **sizeof**(buffer4),1,archivoSalida);

cab -> ChunkSize = (buffer4[0] | (buffer4[1]<<8) | (buffer4[2]<<16) | (buffer4[3]<<24));

printf("(5-8) ChunkSize: %u\n\n",cab -> ChunkSize);

//Formato "Fmt"

lectura = fread(cab -> Format, **sizeof**(cab -> Format),1,archivoEntrada);

printf("(9-12) Format: %s\n\n",cab -> Format);

escritura = fwrite (cab -> Format, **sizeof** (cab -> Format), 1, archivoSalida);

//SubChunk1ID Formato de datos "fmt"

lectura = fread(cab -> SubChunk1ID,**sizeof**(cab -> SubChunk1ID),1,archivoEntrada);

printf("(13-16) SubChunk 1 ID: %s\n\n",cab -> SubChunk1ID);

escritura = fwrite (cab -> SubChunk1ID, **sizeof** (cab -> SubChunk1ID), 1, archivoSalida);

//SubChunk1Size

lectura = fread(buffer4, **sizeof**(buffer4),1,archivoEntrada);

escritura = fwrite(buffer4, **sizeof**(buffer4),1,archivoSalida);

cab -> SubChunk1Size = buffer4[0] | (buffer4[1]<<8) | (buffer4[1]<<16) | (buffer4[1]<<24);

printf("(17-20) SubChunk 1 Size: %u\n\n",cab -> SubChunk1Size);

// Formato de audio

lectura = fread(buffer2, **sizeof**(buffer2),1,archivoEntrada);

escritura = fwrite(buffer2, **sizeof**(buffer2),1,archivoSalida);

cab -> AudioFormat = buffer2[0] | (buffer2[1]<<8);

**if** (cab -> AudioFormat == 1)

strcpy(formatoArchivo,"PCM");

printf("(21-22) Audio Format: %u,%s\n\n",cab -> AudioFormat,formatoArchivo);

//Canales

lectura = fread(buffer2, **sizeof**(buffer2),1,archivoEntrada);

escritura = fwrite(buffer2, **sizeof**(buffer2),1,archivoSalida);

cab -> NumChannels = buffer2[0] | (buffer2[1]<<8);

**if** (cab -> NumChannels == 1)

strcpy(formatoArchivo,"Mono");

**else**

strcpy(formatoArchivo,"Stereo");

printf("(23-24) Number of Channels: %u, Tipo: %s\n\n",cab -> NumChannels,formatoArchivo);

//SampleRate

lectura = fread(buffer4, **sizeof**(buffer4),1,archivoEntrada);

escritura = fwrite(buffer4, **sizeof**(buffer4),1,archivoSalida);

cab -> SampleRate = buffer4[0] | (buffer4[1]<<8) | (buffer4[2]<<16) | (buffer4[3]<<24);

printf("(25-28) Sample Rate: %u\n\n",cab -> SampleRate);

//ByteRate

lectura = fread(buffer4, **sizeof**(buffer4),1,archivoEntrada);

escritura = fwrite(buffer4, **sizeof**(buffer4),1,archivoSalida);

cab -> ByteRate = buffer4[0] | (buffer4[1]<<8) | (buffer4[2]<<16) | (buffer4[3]<<24);

printf("(29-32) Byte Rate: %u BitRate: %u\n\n",cab -> ByteRate,cab -> ByteRate\*8);

//Block Align

lectura = fread(buffer2, **sizeof**(buffer2),1,archivoEntrada);

escritura = fwrite(buffer2, **sizeof**(buffer2),1,archivoSalida);

cab -> BlockAlign = buffer2[0] | (buffer2[1]<<8);

printf("(33-34) Block Align: %u\n\n",cab -> BlockAlign);

//Bits per Sample

lectura = fread(buffer2, **sizeof**(buffer2),1,archivoEntrada);

escritura = fwrite(buffer2, **sizeof**(buffer2),1,archivoSalida);

cab -> BitsPerSample = buffer2[0] | (buffer2[1]<<8);

printf("(35-36) Bits Per Sample: %u\n\n",cab -> BitsPerSample);

//SubChunk2ID

lectura = fread(cab -> SubChunk2ID,**sizeof**(cab -> SubChunk2ID),1,archivoEntrada);

printf("(37-40) SubChunk 2 ID: %s\n\n",cab -> SubChunk2ID);

escritura = fwrite(cab -> SubChunk2ID,**sizeof**(cab -> SubChunk2ID),1,archivoSalida);

//SubChunk2Size

lectura = fread(buffer4, **sizeof**(buffer4),1,archivoEntrada);

escritura = fwrite(buffer4, **sizeof**(buffer4),1,archivoSalida);

cab -> SubChunk2Size = buffer4[0] | (buffer4[1]<<8) | (buffer4[2]<<16) | (buffer4[3]<<24);

printf("(41-44) SubChunk 2 Size: %u\n\n",cab -> SubChunk2Size);

}

**Volumen.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include "Cabecera.c"

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

{

FILE \* archivoEntrada, \* archivoSalida;

cabecera cab;

**int** i, lectura, escritura;

**short** muestra;

**char** \* nombreModificado = (**char** \*) malloc (**sizeof** (**char**));

**char** \* nombreArch = (**char** \*) malloc (**sizeof** (**char**));

system ("cls");

**if** (argc < 3)

printf("Error, faltan argumentos.\n");

**else**

{

nombreArch = (**char** \*) argv [1];

nombreModificado = (**char** \*) argv [2];

}

//Abrimos los archivos en modo binario

archivoEntrada = abreArchivo (nombreArch, nombreModificado,1);

archivoSalida = abreArchivo (nombreArch, nombreModificado, 2);

//Leemos e imprimimos la cabecera del archivo wav

leerCabecera (archivoEntrada, archivoSalida, &cab);

**for** (i = 0; i < (cab.SubChunk2Size); i ++)

{

lectura = fread(&muestra, **sizeof** (**short**), 1, archivoEntrada);

muestra \*= 0.5; //Dividimos a la mitad cada uno de los datos

escritura = fwrite(&muestra, **sizeof** (**short**), lectura, archivoSalida); //Escribimos los datos nuevos en el archivo

}

printf ("\n\n");

fclose (archivoEntrada);

fclose (archivoSalida);

printf ("Archivo '%s' modificado correctamente.\n", nombreArch);

**return** 0;

}