**Septiembre**

**20**

13

**Ramos Oromi - Stessens**

Trabajo Práctico 1 – Manejo de archivos WAVE

Introducción a la computación - Universidad Austral

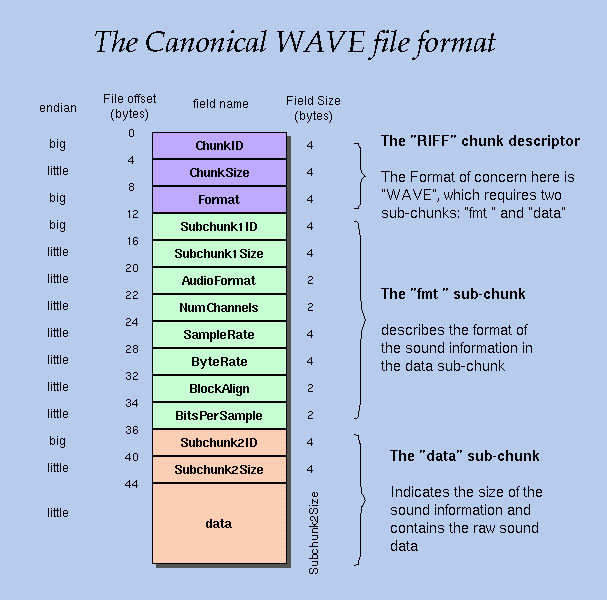
08

**Fall**

# Introducción:

El siguiente trabajo busca demostrar la composición de un archivo de audio, ahondando en el audio digital. Trabajamos con el formato WAVE, y pudimos comprender la composición y el mapeo de una señal de audio a un archivo.

# Marco teórico:

El formato de archivos WAVE es un subset de la especificación RIFF de Microsoft, y esta compuesto como se ve a continuación:

Lo que buscamos en este trabajo es, tomando un archivo WAVE sin compresión alguna, descomponerlo y generar un archivo nuevo, que sea el original pero mas rápido.

# Desarrollo:

Realizamos un código en Java, que se encarga de desarmar el WAVE y de generar un archivo nuevo. El código en cuestión es el que se ve a continuación:

public class Filtro {

public static void main(String[] args) {

int tamanioFormatChunk = 0;

int bytesExtra = 0;

int indiceData;

int tamanioDataChunk = 0;

int comienzoDatos;

boolean estaComprimido;

int tamanioArchivo;

int formatoDeAudio = 0;

int canalesDeSalida = 0;

int frecuencia = 0;

int bitsPorSegundo = 0;

double a1 = 1, a2 = 0, a3 = 0, a4 = 0, a5 = 0, b0 = 1, b1 = 0, b2 = 0, b3 = 0, b4 = 0, b5 = 0;

try {

Path path = Paths.get("uno.wav");

byte[] data = Files.readAllBytes(path);

if (("" + (char) data[0] + (char) data[1] + (char) data[2] + (char) data[3]).equals("RIFF"))

System.out.println("Es un archivo RIFF");

tamanioArchivo = (data[4] + (data[5] \* 256) + (data[6] \* 65536) + (data[7] \* 16777216));

if (("" + (char) data[8] + (char) data[9] + (char) data[10] + (char) data[11]).equals("WAVE"))

System.out.println("Es un archivo WAV");

if (("" + (char) data[12] + (char) data[13] + (char) data[14]).equals("fmt")) {

tamanioFormatChunk = (data[16] + (data[17] \* 256) + (data[18] \* 65536) + (data[19] \* 16777216));

System.out.println("Tamaño de format chunk: " + tamanioFormatChunk);

formatoDeAudio = (data[20] + (data[21] \* 256));

if (formatoDeAudio == 1) {

estaComprimido = false;

System.out.println("El archivo no esta comprimido");

} else {

estaComprimido = true;

System.out.println("El archivo esta comprimido");

}

canalesDeSalida = (data[22] + (data[23] \* 256));

System.out.println("Cantidad de canales de salida: " + canalesDeSalida);

frecuencia = (data[24] + (data[25] \* 256) + (data[26] \* 65536) + (data[27] \* 16777216));

System.out.println("Frecuencia: " + frecuencia);

bitsPorSegundo = (data[34] + (data[35] \* 256));

System.out.println("bPS: " + bitsPorSegundo);

if (estaComprimido) {

bytesExtra = (data[36] + (data[37] \* 256));

System.out.println("Bytes de más: " + bytesExtra);

}

}

indiceData = 19 + tamanioFormatChunk + bytesExtra + 1;

if (("" + (char) data[indiceData] + (char) data[indiceData + 1] + (char) data[indiceData + 2] +

(char) data[indiceData + 3]).equals("data")) {

tamanioDataChunk = (data[indiceData + 4] + (data[indiceData + 5] \* 256) +

(data[indiceData + 6] \* 65536) + (data[indiceData + 7] \* 16777216));

System.out.println("Tamaño de data chunk: " + tamanioDataChunk);

}

comienzoDatos = indiceData + 8;

DataOutputStream outFile = new DataOutputStream(new FileOutputStream("filtrado.wav"));

int bytesPorSegundo = frecuencia \* canalesDeSalida \* bitsPorSegundo / 8;

int bytesPorMuestra = canalesDeSalida \* bitsPorSegundo / 8;

// write the wav file per the wav file format

outFile.writeBytes("RIFF"); // 00 - RIFF

tamanioArchivo = tamanioArchivo - tamanioDataChunk / 2;

outFile.write(ByteBuffer.allocate(4).order(ByteOrder.LITTLE\_ENDIAN).putInt(tamanioArchivo).array()); // 04 - how big is the rest of this file?

outFile.writeBytes("WAVE"); // 08 - WAVE

outFile.writeBytes("fmt "); // 12 - fmt

outFile.write(ByteBuffer.allocate(4).order(ByteOrder.LITTLE\_ENDIAN).putInt(tamanioFormatChunk).array()); // 16 - size of this chunk

outFile.write(ByteBuffer.allocate(2).order(ByteOrder.LITTLE\_ENDIAN).putShort((short) formatoDeAudio).array()); // 20 - what is the audio format? 1 for PCM = Pulse Code Modulation

outFile.write(ByteBuffer.allocate(2).order(ByteOrder.LITTLE\_ENDIAN).putShort((short) canalesDeSalida).array()); // 22 - mono or stereo? 1 or 2? (or 5 or ???)

outFile.write(ByteBuffer.allocate(4).order(ByteOrder.LITTLE\_ENDIAN).putInt(frecuencia).array()); // 24 - samples per second (numbers per second)

outFile.write(ByteBuffer.allocate(4).order(ByteOrder.LITTLE\_ENDIAN).putInt(bytesPorSegundo).array()); // 28 - bytes per second

outFile.write(ByteBuffer.allocate(2).order(ByteOrder.LITTLE\_ENDIAN).putShort((short) bytesPorMuestra).array()); // 32 - # of bytes in one sample, for all channels

outFile.write(ByteBuffer.allocate(2).order(ByteOrder.LITTLE\_ENDIAN).putShort((short) bitsPorSegundo).array()); // 34 - how many bits in a sample(number)? usually 16 or 24

outFile.writeBytes("data"); // 36 - data

System.out.println("Nuevo tamaño data chunk: " + tamanioDataChunk);

outFile.write(ByteBuffer.allocate(4).order(ByteOrder.LITTLE\_ENDIAN).putInt(tamanioDataChunk).array()); // 40 - how big is this data chunk

DataInputStream inFile = new DataInputStream(new FileInputStream("uno.wav"));

for (int i = 0; i < comienzoDatos; i++) {

inFile.read();

}

byte[] nuevaData = new byte[tamanioDataChunk];

int n = comienzoDatos + 6;

for (int i = 6; i < tamanioDataChunk; i++) {

nuevaData[i] = (byte) (b0 \* data[n] + b1 \* data[n - 1] + b2 \* data[n - 2] + b3 \* data[n - 4] +

b4 \* data[n - 5] + b5 \* data[n - 6] - a1 \* nuevaData[i - 1] - a2 \* nuevaData[i - 2] -

a3 \* nuevaData[i - 3] - a4 \* nuevaData[i - 4] - a5 \* nuevaData[i - 6]);

}

ByteBuffer buffer = ByteBuffer.wrap(nuevaData);

outFile.write(buffer.array()); // 44 - the actual data itself - just a long string of numbers

} catch (IOException e) {

e.printStackTrace();

}

}

}

Cuando ejecutamos ese código, se genera un archivo WAVE nuevo, que corresponde a la versión rápida del sonido.