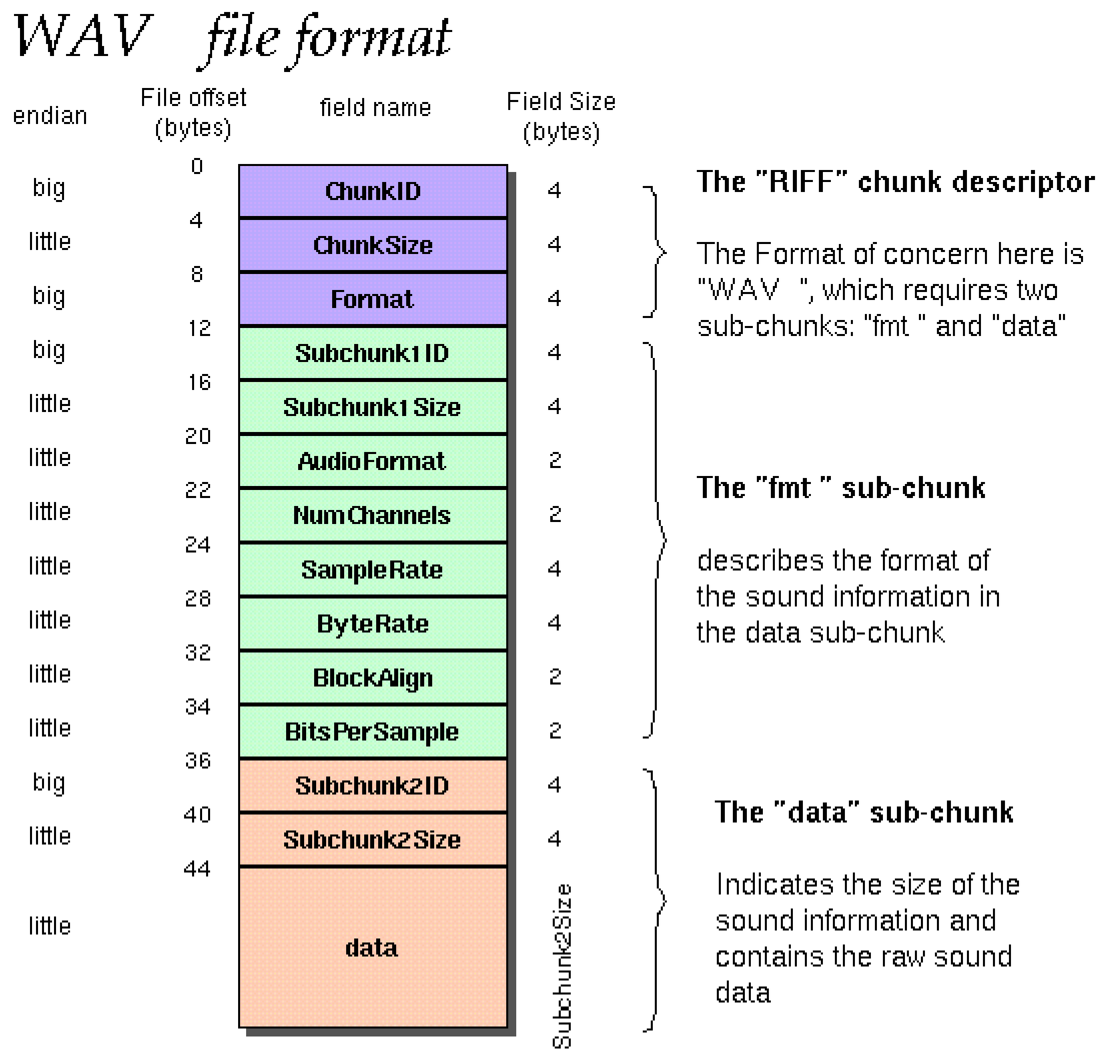
ECE 3331, Dr. Hebert

WAV soundfile format

Audio file formats are used to store audio data on a computer system, mp3 player, DVD, etc.
There are many types of audio file formats. These include the MP3, WAV, AIFF, AU, RA formats.
Each format has certain strengths and weaknesses.
The WAV format, designed by Microsoft, will be used for your project. Files stored in WAV format must be stored in a file named with the file_extension *.wav .

The WAV file format is a subset of Microsoft's RIFF specification for the storage of multimedia files. A RIFF file starts out with a file header followed by a sequence of data chunks as follows. A WAV file is typically just a RIFF file which consists of three chunks – (1) a chunk indicating the the file is a RIFF WAV file. (2) a header consisting of a "fmt " chunk that specifies the data format e.g. stereo, 44100 samples per sec, etc.. and (3) a "data" chunk containing the actual sample data stored in the format that has been indicated in the “fmt “ chunk. A WAV PCM file is one that is stored in uncompressed format. PCM stands for pulse-code-modulation which here simply refers to uncompressed mode.

The overview below, shows the WAV file format with various data "chunks" and data "sub-chunks".
Each chunk or sub-chunk consists of a name (RIFF, fmt, data, or other), a size of the chunk or sub-chunk in bytes, followed by the data itself, such that the total chunk or sub-chunk is of the size specified. The WAV format allows various other data"sub-chunks" not listed below, that contain information not necessary for playing the audio file. Only RIFF, fmt, data are necessary for playing an audio file. When your program reads the name of a chunk or sub-chunk (RIFF, fmt, data, or other), if the name is not one of those necessary for playing the audio file (RIFF, fmt, data), your program should read the size of the chunk or sub-chunk, then skip over the appropriate number of bytes using fseek( ).


WAV file format
endian
File offset (bytes)
field name
Field Size (bytes)
big 0 ChunkID 4
little 4 ChunkSize 4
big 8 Format 4
big 12 Subchunk1ID 4
little 16 Subchunk1 Size 4
little 20 AudioFormat 2
little 22 NumChannels 2
little 24 SampleRate 4
little 28 ByteRate 4
little 32 BlockAlign 2
little 34 BitsPerSample 2
big 36 Subchunk2ID 4
little 40 Subchunk2 Size 4
little 44 data Subchunk2Size
The "RIFF" chunk descriptor
The Format of concern here is "WAVE", which requires two sub-chunks: "fmt " and "data"
The "fmt " sub-chunk
describes the format of the sound information in the data sub-chunk
The "data" sub-chunk
Indicates the size of the sound information and contains the raw sound data

The WAV format starts with the RIFF chunk:

Offset	Size	Name	Description
0	4	**ChunkID**	Contains the letters "RIFF" in ASCII form (0x52494646 big-endian form). Read this into your program as **unformatted** 4-characters
4	4	**ChunkSize**	36 + SubChunk2Size, or more precisely: 4 + (8 + SubChunk1Size) + (8 + SubChunk2Size) (unsigned 4-byte int stored LSB 1st) This is the size of the rest of the chunk following this stored number. This is the size of the entire file in bytes minus 8 bytes for the two fields not included in this count: ChunkID and ChunkSize. Read this into your program as an **unformatted** unsigned int.
8	4	**Format**	Contains the letters "WAVE" (0x57415645 big-endian form). Read this value into your program as **unformatted** 4-characters

The "WAV" format consists of two subchunks: "fmt " and "data":
The "fmt " subchunk describes the audio data's format:

Offset	Size	Name	Description
12	4	**Subchunk1ID**	Contains the letters "fmt " (0x666d7420 big-endian form). Read this value into your program as unformatted 4-characters
16	4	**Subchunk1Size**	16 . (4-byte unsigned int stored LSB 1st) This is the size of the remaining part of the Subchunk “fmt “ that follows **Subchunk1Size**; i.e. the size of subchunk “fmt “ minus 8. The 4 bytes are ordered LSB 1st. Read this value in as an unformatted unsigned int.
20	2	**AudioFormat**	1 (for PCM i.e. uncompressed) (unformatted 2-byte unsigned int stored LSB 1st) Values other than 1 indicate various forms of audio compression. Read this in as an unformatted unsigned short int.
22	2	**NumChannels**	Mono = 1, Stereo = 2, etc. (unsigned 2-byte int stored LSB 1st) Read this in as an unformatted unsigned short int.
24	4	**SampleRate**	8000, 11025, or 22050, or 44100, etc. samples per second. (unsigned 4-byte int stored LSB 1st) Read this value in as an unformatted unsigned int.
28	4	**ByteRate**	(bytes per second) = SampleRate * NumChannels * BitsPerSample/8 (unsigned 4-byte int stored LSB 1st) Read this value in as an unformatted unsigned int.
32	2	**BlockAlign**	== NumChannels * BitsPerSample/8 (stored as LSB 1st) The number of bytes for one "audio sample" (including all channels). This number need not be used by the audioplayer, as it can compute this value itself, but it provides a check on the correctness of the audio file itself. Use fread( ) to read this value as an unformatted unsigned short int.
34	2	**BitsPerSample**	8 bits = 8, 16 bits = 16, etc. (unsigned 2-byte int stored LSB 1st)

If the file is not in uncompressed (PCM) form, additional parameters are stored in a pre-defined order after this.
The wav files that we will work with will be in uncompressed form.

The "data" chunk contains the size of the data and the actual sound:

Offset	Size	Name	Description
36	4	**Subchunk2ID**	Contains the letters "data" (0x64617461 big-endian form).
40	4	**Subchunk2Size**	== NumSamples * NumChannels * BitsPerSample/8 (stored as LSB 1st)

This is the number of bytes in the data.
You can also think of this as the size of the read-statement of the subchunk following this number.

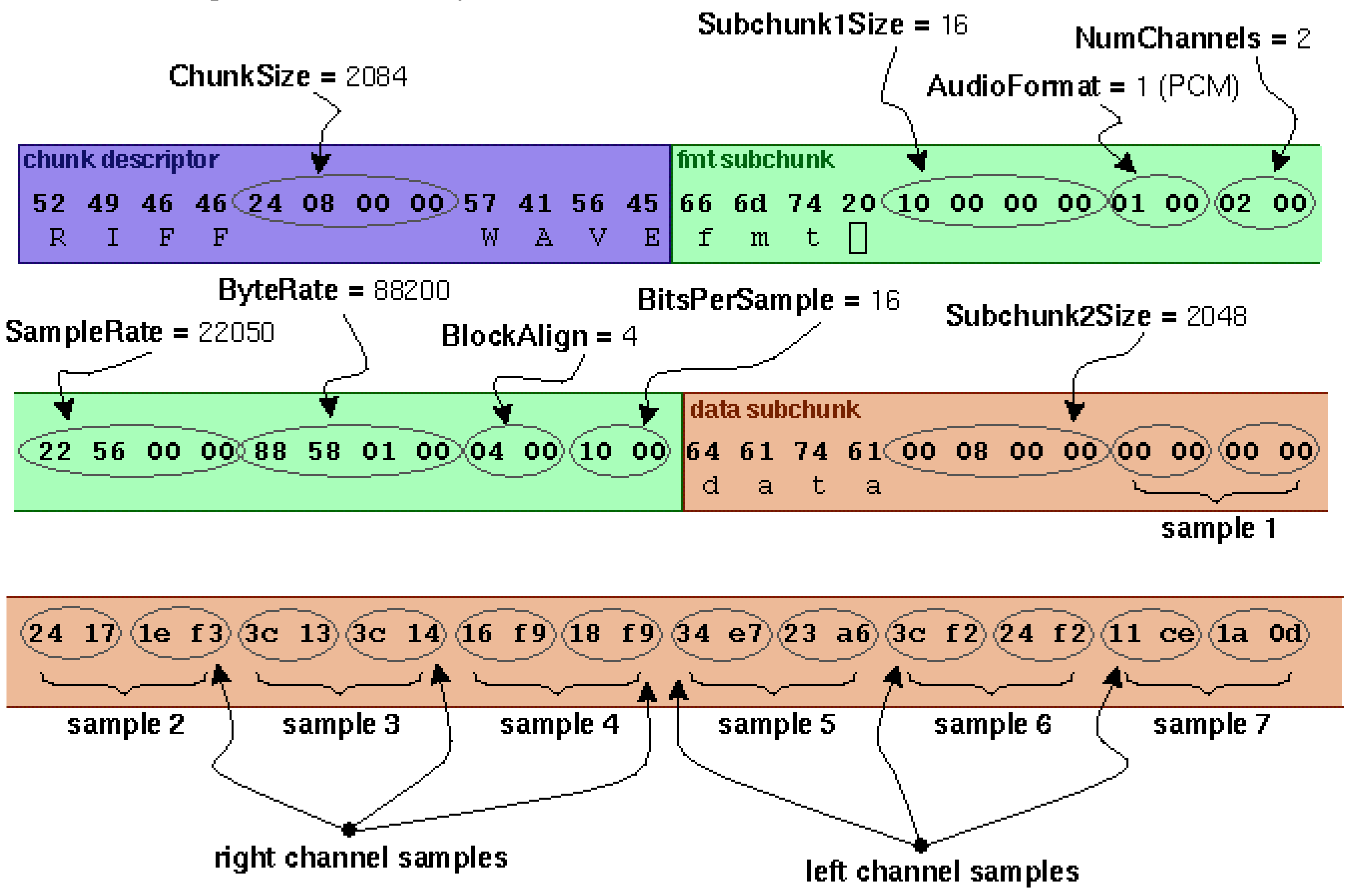
44 * **Data** The actual audio data. If stereo, it is stored as left-channel sample, right-channel sample, left-channel sample, right-channel sample, left-channel sample, etc.

As an example, here are the first 72 bytes of a WAV file with bytes shown

as hexadecimal 1-byte 00 to FF numbers:
52 49 46 46 24 08 00 00 57 41 56 45 66 6d 74 20 10 00 00 00 01 00 02 00
22 56 00 00 88 58 01 00 04 00 10 00 64 61 74 61 00 08 00 00 00 00 00 00
24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d

As decimal 1-byte 0-255 numbers:
82 73 70 70 36 8 0 0 87 65 66 69 etc

Here is the interpretation of these bytes as a WAV soundfile:



For this example, there would be 2048 bytes of audio data following the Subchunk2Size number = 2048.
Since there are 2-channels here, the audio data would be 1024 bytes of audio data for the left channel, 1024 bytes of audio data for the right channel.
In this example, each sample is 2-bytes, so we would have 512 samples for the left channel, 512 samples for the right channel.
Since the SampleRate = 22050 (samples per second), we have (512 / 22050) seconds of audio in this file.
.