Structure of a .wav file

The .wav file format is derived from the Resource Interchange File Format (.riff) specification, which includes multimedia data. The structure of a .riff file is based on chunks and subchunks. Each chunk and subchunk has a type that is represented by a four-character tag in big-endian byte order. Multiple kinds of data can be stored in a .riff file, but we describe the canonical .wav file format here. The first element in the file is the .riff chunk descriptor, which describes the .riff file that we have.

The .riff chunk descriptor:

Field name and description	Field size in bytes	Endian byte order
Chunk identifier: This field contains the letters "RIFF" in ASCII form. Each character is 1 byte.	4	Big
Chunk size: This field contains the size of the memory chunk that appears immediately after this field. The total file size is the value of this field plus 8 bytes (for the chunk identifier and chunk size).	4	Little
Format: This field contains the letters "WAVE" in ASCII form. Each character is 1 byte.	4	Big

Format subchunk:

Because the format value of the .riff file is specified as "WAVE", we know that this .riff file contains .wav data only. The .wav file format requires that the format and data subchunks be handled.
The format subchunk appears after the .riff chunk descriptor. The format subchunk describes metadata about the .wav file, such as sample rate, sample width and is structured as follows:

Field name and description	Field size in bytes	Endian byte order
Subchunk identifier: This field contains the characters "fmt " (including the space) in ASCII form. Each character is 1 byte.	4	Big
Subchunk size: This field contains the size, in bytes, of the remaining format subchunk that appears immediately after this field, which is 16 bytes for pulse code modulation (PCM). The total format subchunk size is this value plus 4 bytes (for the subchunk identifier).	4	Little
Audio format: This field determines the audio format. For a .wav file, the value is 1. Values other than 1 indicate some form of compression.	2	Little
Number of channels: This field is the number of separate streams of audio information. For example, if you use audio data for one channel (mono), the value is 1.	2	Little
Sample rate: This field is the number of samples per unit of time, typically in seconds.	4	Little
Byte rate: This field is the number of bits per sample	4	Little

divided by eight (8 bits per byte) and multiplied by the sample rate.		
Block align: This field is the number of bits per sample divided by eight and multiplied by the number of channels.	2	Little
Bits per sample: This field is the number of bits used for each sample. For example, if you use 8 bits per sample, the value of this field is 8.	2	Little
In PCM format, there is nothing at the end of this subchunk, but in non-PCM format, there is space for extra parameters. In PCM format, this space is empty, but it must be skipped to read the file correctly.	-	-

Data subchunk:

After the format subchunk is the data subchunk. The data subchunk contains the size of the audio data (in bytes) and the sound data. Here is the structure of the data subchunk:

Field name and description	Field size in bytes	Endian byte order
Subchunk identifier: This field contains the letters "data" in ASCII form. Each character is 1 byte.	4	Big
Subchunk size: This field contains the size of the audio data that appears immediately after this field. This data size can also be considered the number of bits per sample divided by eight, multiplied by the	4	Little

number of samples, and multiplied by the number of channels.		
Data: This field represents audio data. The data is read based on the metadata obtained above.	-	Little

NOTE ON DATA CHUNK:

For the sample data, it's important to note that each element of the data array is a signed value – it can be negative, positive or zero. The range of each of these elements is important too.
Since each sample represents amplitude, and we are working with signed values, we have to consider what is our minimum and maximum amplitude given the data type we've chosen. Due to some crazy business involving Endianness and 2's complement, **16-bit samples range from -32760 to 32760** instead of -32768 to 32768 (2^16 / 2).

MAKING A WAV FILE

Writing the wave file is as easy as constructing the Header, Format Chunk and Data Chunk and writing them in binary fashion to a file in that order.

GOOD READS

How is audio represented with numbers?