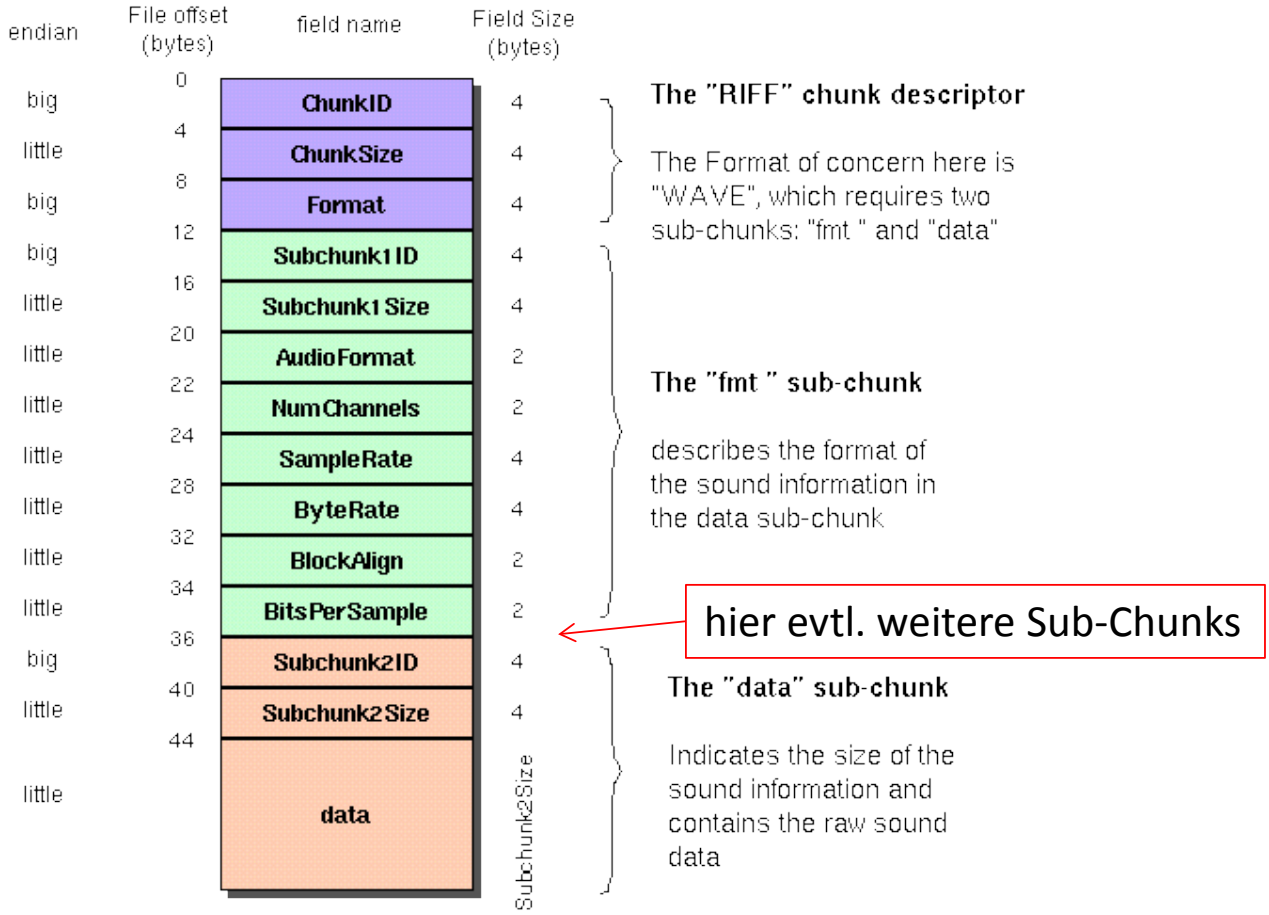
3. Übung zu TEI2_CPR

1. Einlesen des Headers des Audio-Dateiformats WAV

Das Audio-Dateiformat WAV (Dateiendung .wav) ist ein Spezialfall des *Resource Interchange File Format* (RIFF).
RIFF (und damit WAV) setzt sich aus sog. *Chunks* (Stücken) zusammen, die wiederum Chunks (Sub-Chunks) enthalten können.
Jeder Chunk besteht aus den folgenden drei Teilen:

- ChunkID, 4 Zeichen (char): Typ des chunks. Die erste ChunkID muss „RIFF“ sein.
- ChunkSize, unsigned int (4 Bytes, little endian): Größe der Daten des Chunks.
- Daten (Anzahl Bytes gemäß ChunkSize): Inhalt je nach ChunkID.

Die folgende Abbildung (Quelle: http://soundfile.sapp.org/doc/WaveFormat/) zeigt wie dieses Konzept bei WAV (.wav) verwendet wird.



Bemerkungen:

- Zum Chunk „RIFF“:
  - ChunkSize: Gesamt-Größe aller Daten. Diese ist um 8 kleiner als die Dateigröße, da die Chunk-Beschreibung (bestehend aus ChunkID und die ChunkSize, also 2x4 Bytes) nicht zu den Daten des Chunks gehört.
  - Daten: die ersten 4 Bytes sind das „Format“ (siehe Abbildung, unterster blauer Eintrag), bestehend aus 4 Zeichen (char), die im Falle von WAV auf „WAVE“ gesetzt sind.
  - Format „WAVE“ bedeutet, dass der Rest der Datei aus Sub-Chunks besteht. Zwei weitere Sub-Chunks sind verpflichtend: „fmt “ (fmt und ein Leerzeichen) und „data“.
- Der Sub-Chunk „fmt “:
  - Typischerweise der erste Sub-Chunk.
  - ChunkSize: typischerweise 16 – vollziehen Sie nach anhand der Abbildung nach, wieso die Größe hier 16 ist.

    - Wir wollen uns auf Dateien mit dieser typischen Größe des Sub-Chunk „fmt “ beschränken.
    - Daten: mindestens die in grün dargestellten Felder (ab „AudioFormat“).
        - „AudioFormat“: Art der Codierung der Audio-Information (Abtastwerte) im Sub-Chunk „data“. Wir wollen uns hier auf die Codierung „3“ (IEEE float) beschränken.
        - „NumChannels“: Anzahl der Kanäle (1=Mono, 2=Stereo). Wir wollen uns auf Mono beschränken.
        - „SampleRate“: Anzahl der Audio-Abtastwerte pro Sekunde.
- Bis hierhin ist die Datenstruktur in der Struktur `wavheader` aus `wave.h` abgebildet.
- Es folgen weitere Sub-Chunks, die wieder jeweils mit ChunkID und ChunkSize (abgebildet als Struktur `chunkheader` aus `wave.h`) beginnen. ChunkSize ist wieder die Größe der danach folgenden Daten des Sub-Chunks in Bytes.
- Zum Sub-Chunk „data“:
    - Enthält als Daten die eigentliche Audio-Daten (Abtastwerte).
    - Nicht unbedingt direkt nach dem Sub-Chunk „fmt “. Oft der letzte Sub-Chunk.

Auf der Moodle-Seite befinden Sie die Datei `wave.h`, die in der Struktur `wavheader` den Chunk-Header von „RIFF“ und „fmt “ abbildet.

- Erstellen Sie ein neues Projekt „Audio1“.
- Lesen Sie den Datei-Header (Sub-Chunks „RIFF“ und „fmt “) der Datei `test.wav` (von der Moodle-Seite) in eine Variable der Struktur `wavheader` ein.
- Lassen Sie sich die folgenden Felder zur Kontrolle und zum Verständnis des Dateiinhalts ausgeben:
    - ChunkID von „RIFF“ und „fmt “.
    - ChunkSize von „RIFF“. Vergleichen Sie diese mit der Größe der Datei!
    - ChunkSize von „fmt “. Entspricht der Wert der Erwartung?
    - AudioFormat. Oben steht, dass wir uns auf ein Format beschränken wollen. Hat die Datei das richtige AudioFormat? Welcher Wert muss dann hier stehen?
    - Ist es eine Mono-Datei? Lassen Sie sich den Wert ausgeben, in dem das steht.
    - Wie viele Audio-Abtastwerte pro Sekunde enthält die Datei?
- Berechnen Sie, wie viele Sekunden die Aufnahme lang ist. Kontrollieren Sie das Ergebnis mit einem Audio-Player.
- Nach dem Sub-Chunk „fmt “ folgen weitere Sub-Chunks. Schreiben Sie eine Funktion `readSubChunks(...)`, die den Chunk-Header einliest, die ChunkID anzeigt, die eigentlichen Daten des Sub-Chunks überspringt (C-Funktion `fseek`) und so lange damit vorfährt, bis die Datei zu Ende ist (C-Funktion `feof`).

2. Einlesen der Audio-Daten (Abtastwerte)

Ändern Sie Ihre Funktion `readSubChunks(...)` zu einer Funktion `readDataChunk(...)`, die:

- den Sub-Chunk „data“ in der Datei sucht,
- die Audio-Daten in ein dynamisch reserviertes Array von `float`-Werten einliest und
- dieses Array und die Anzahl der eingelesenen Werte zurückliefert (da in C wie in Java nur ein Rückgabewert möglich ist, muss mind. eine Information über einen Parameter zurückgeliefert werden).
- Probieren Sie aus, ob das funktioniert hat, indem Sie die eingelesenen Daten mit Hilfe der Funktion `writePCM` (in `wave.c` auf der Moodle-Seite) wieder abspeichern und die abgespeicherte Datei in einem Audio-Player abspielen lassen.

3. Audio-Daten schneller abspielen

Schreiben Sie eine Funktion `schneller(...)`, die jeden zweiten Abtastwert in ein neues Array speichert. Speichern Sie die Datei mit Hilfe der Funktion `writePCM` ab (in `wave.c` auf der Moodle-Seite).