20
0

@syuhei1008 2016年01月24日に更新

WAVEファイル読み・書き込み

C Sound

⚠ この記事は最終更新日から3年以上が経過しています。

WAVEファイルの読み込みと書き込みをC言語を用いて行う。
参考文献:C言語で始める音のプログラミング

WAVEファイルについて

フォーマット

RIFF WAVE File Format				
Endian	File Offset	Field Name	Field Size	Description
big	0	Chunk ID	4	"RIFF" Chunk Descriptor The format field here is "WAVE", and it requires two sub-chunks: "fmt" chunk and "data" chunk.
little	4	Chunk Size	4	
big	8	Format	4	
big	12	SubChunk1 ID	4	
little	16	SubChunk1 Size	4	
little	20	Audio Format	2	"fmt" Sub-Chunk This region contains all essential information we need to know about a WAVE file, such as sample rate, channel number etc.
little	22	Channel Number	2	
little	24	Sample Rate	4	
little	28	Byte Rate	4	
little	32	Block Align	2	
little	34	Bits Per Sample	2	
big	36	SubChunk2 ID	4	"data" Sub-Chunk The field "SubChunk2 Size" indicates the size of RAW audio data.
little	40	SubChunk2 Size	4	
little	44	Data	SubChunk2 Size	

WAVEファイルはRIFF(Resource Interchange File Format)チャンクと呼ばれるブロック構造により、音のデータを記録している。

RIFFチャンクの中にfmtチャンクとdataチャンクが含まれている。

- fmtチャンク：標本化周波数や量子化精度など音データのプロパティを格納
- dataチャンク：音データそのもの

プログラム

400Hzの波形を出力する音声ファイルを作成し、それを読み込むプログラム

```
lang:main.c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "wave.h"

int main(int argv, char **argc)
{
    MONO_PCM pcmForWrite, pcmForRead;
    int i;
    double amplitude, frequency;

    pcmForWrite.fs = 8000;
```

```
    pcmForWrite.bits = 16;
    pcmForWrite.length = 8000;
    pcmForWrite.s = calloc(pcmForWrite.length, sizeof(double));

    amplitude = 0.25;
    frequency = 400.0;

    for(i = 0; i < pcmForWrite.length; i++)
    {
        pcmForWrite.s[i] = amplitude * sin((2.0 * M_PI * frequency * i) / pcmForWrite.fs);
        // A sin(2πft0/fs) + A/2 sine(2πft1/fs) ・・・
    }
    monoWaveWrite(&pcmForWrite, "testWave.wav");
    free(pcmForWrite.s);

    monoWaveRead(&pcmForRead, "testWave.wav");
    for(i = 0; i < pcmForRead.length; i++)
    {
        printf("%d:%lf\n", i, pcmForRead.s[i]);
    }
    free(pcmForRead.s);

    return 0;
}
```

```
lang:wave.h
#ifndef WAVE_H
#define WAVE_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define GROUND 32768.0
#define MAXPLUS 65536.0
#define MAXMINUS 0.0

typedef struct
{
    int fs;
    // 標本化周波数
    int bits;
    // 量子化制度
    int length;
    // データ長
    double *s;
    // 音データ
} MONO_PCM;
// モノラルの音声データの定義

typedef struct
{
    char chunkID[4];
    long chunkSize;
    char chunkFormType[4];
} RIFF_CHUNK;
// RIFFチャンクの定義

typedef struct
{
    char chunkID[4];
    long chunkSize;
    short waveFormatType;
    short formatChannel;
    long samplesPerSec;
    long bytesPerSec;
    short blockSize;
    short bitsPerSample;
} FMT_CHUNK;
// fmtチャンクの定義

typedef struct
{
```

```
    char chunkID[4];
    long chunkSize;
    short data;
} DATA_CHUNK;
// dataチャンクの定義

typedef struct
{
    RIFF_CHUNK riffChunk;
    FMT_CHUNK fmtChunk;
    DATA_CHUNK dataChunk;
} WAVE_FORMAT;
// WAVEフォーマット

void monoWaveRead(MONO_PCM *pcm, char *fileName);
// モノラルの音声データ(waveファイル)の読み込み

void monoWaveWrite(MONO_PCM *pcm, char *fileName);
// モノラルの音声データ(waveファイル)書き込み

#endif
```

wave.c

```
#include "wave.h"

void monoWaveRead(MONO_PCM *pcm, char *fileName)
{
    FILE *fp;
    int i;
    WAVE_FORMAT waveFormat;

    fp = fopen(fileName, "rb");

    if(!fp)
    {
        printf("file open error");
        exit(0);
    }

    fread(waveFormat.riffChunk.chunkID, 1, 4, fp);
    fread(&waveFormat.riffChunk.chunkSize, 4, 1, fp);
    fread(waveFormat.riffChunk.chunkFormType, 1, 4, fp);
    // RIFFチャンクの読み込み

    fread(waveFormat.fmtChunk.chunkID, 1, 4, fp);
    fread(&waveFormat.fmtChunk.chunkSize, 4, 1, fp);
    fread(&waveFormat.fmtChunk.waveFormatType, 2, 1, fp);
    fread(&waveFormat.fmtChunk.formatChannel, 2, 1, fp);
    fread(&waveFormat.fmtChunk.samplesPerSec, 4, 1, fp);
    fread(&waveFormat.fmtChunk.bytesPerSec, 4, 1, fp);
    fread(&waveFormat.fmtChunk.blockSize, 2, 1, fp);
    fread(&waveFormat.fmtChunk.bitsPerSample, 2, 1, fp);
    // fmtチャンクの読み込み

    fread(waveFormat.dataChunk.chunkID, 1, 4, fp);
    fread(&waveFormat.dataChunk.chunkSize, 4, 1, fp);

    pcm->fs = waveFormat.fmtChunk.samplesPerSec;
    pcm->bits = waveFormat.fmtChunk.bitsPerSample;
    pcm->length = waveFormat.dataChunk.chunkSize;
    pcm->s = calloc(pcm->length, sizeof(double));

    short data;
    for(i = 0; i < pcm->length; i++)
    {
        fread(&data, 2, 1, fp);
        pcm->s[i] = (double)data/GROUND;
    }
    // dataチャンクの読み込み

    fclose(fp);
}

void monoWaveWrite(MONO_PCM *pcm, char *fileName)
```

```
{
    FILE *fp;
    int i;
    WAVE_FORMAT waveFormat;

    strcpy(waveFormat.riffChunk.chunkID, "RIFF");
    waveFormat.riffChunk.chunkSize = 36 + pcm->length * 2;
    strcpy(waveFormat.riffChunk.chunkFormType, "WAVE");

    strcpy(waveFormat.fmtChunk.chunkID, "fmt ");
    waveFormat.fmtChunk.chunkSize = 16;
    waveFormat.fmtChunk.waveFormatType = 1;
    // PCMの場合は1
    waveFormat.fmtChunk.formatChannel = 1;
    // モノラルの場合は1,ステレオの場合は2
    waveFormat.fmtChunk.samplesPerSec = pcm->fs;
    waveFormat.fmtChunk.bytesPerSec = (pcm->fs * pcm->bits) / 8;
    waveFormat.fmtChunk.blockSize = pcm->bits / 8;
    waveFormat.fmtChunk.bitsPerSample = pcm->bits;

    strcpy(waveFormat.dataChunk.chunkID, "data");
    waveFormat.dataChunk.chunkSize = pcm->length * 2;

    fp = fopen(fileName, "wb");

    if(!fp)
    {
        printf("file open error");
        exit(0);
    }

    fwrite(waveFormat.riffChunk.chunkID, 1, 4, fp);
    fwrite(&waveFormat.riffChunk.chunkSize, 4, 1, fp);
    fwrite(waveFormat.riffChunk.chunkFormType, 1, 4, fp);
    // RIFFチャンクの書き込み

    fwrite(waveFormat.fmtChunk.chunkID, 1, 4, fp);
    fwrite(&waveFormat.fmtChunk.chunkSize, 4, 1, fp);
    fwrite(&waveFormat.fmtChunk.waveFormatType, 2, 1, fp);
    fwrite(&waveFormat.fmtChunk.formatChannel, 2, 1, fp);
    fwrite(&waveFormat.fmtChunk.samplesPerSec, 4, 1, fp);
    fwrite(&waveFormat.fmtChunk.bytesPerSec, 4, 1, fp);
    fwrite(&waveFormat.fmtChunk.blockSize, 2, 1, fp);
    fwrite(&waveFormat.fmtChunk.bitsPerSample, 2, 1, fp);
    // fmtチャンクの書き込み

    fwrite(waveFormat.dataChunk.chunkID, 1, 4, fp);
    fwrite(&waveFormat.dataChunk.chunkSize, 4, 1, fp);

    short data;
    double s;
    for(i = 0; i < pcm->length; i++)
    {
        s = ((pcm->s[i] + 1.0) / 2) * (MAXPLUS + 1.0);
        if(s > MAXPLUS)
            s = MAXPLUS;
        else if(s < MAXMINUS)
            s = MAXMINUS;

        data = (short)(s + 0.5) - GROUND;
        fwrite(&data, 2, 1, fp);
    }

    fclose(fp);
}
```

まとめ

- fmtチャンクのchunkIDを書き込む箇所でstrcpyを使っている。strcpyはコピー元の文字列のサイズがコピー先のサイズに満たない場合「¥0」が追記される。これによりWAVEファイルとして認識されず、音が鳴らなかった。

• 次回は様々な波形をプログラムにより作成してみる。

編集リクエスト ストック いいね 20

@syuhei1008

フォロー

ユーザー登録して、Qiitaをもっと便利に使ってみませんか。

1. あなたにマッチした記事をお届けします
ユーザーやタグをフォローすることで、あなたが興味を持つ技術分野の情報をまとめてキャッチアップできます

2. 便利な情報をあとで効率的に読み返せます
気に入った記事を「ストック」することで、あとからすぐに検索できます

より詳しく

登録する ログインする

© 2011-2019 Increments Inc. 利用規約 ガイドライン プライバシー ヘルプ

Qiita とは ユーザー タグ 記事 ブログ API Qiita Jobs Qiita:Team Qiita:Zine 広告掲載 ご意見