#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <memory.h>

#define FS 48000.0f

#define FL 1500.0f

#define FH 3500.0f

#define M 32

#define PI 3.141592653589793f

typedef struct \_wav {

int fs;

char header[44];

size\_t length;

short \*LChannel;

short \*RChannel;

} wav;

int wav\_read\_fn(char \*fn, wav \*p\_wav)

{

//char header[44];

short temp = 0;

size\_t i = 0;

FILE \*fp = fopen(fn, "rb");

if(fp==NULL) {

fprintf(stderr, "cannot read %s\n", fn);

return 0;

}

fread(p\_wav->header, sizeof(char), 44, fp);

while( !feof(fp) ) {

fread(&temp, sizeof(short), 1, fp);

i++;

}

p\_wav->length = i / 2;

p\_wav->LChannel = (short \*) calloc(p\_wav->length, sizeof(short));

if( p\_wav->LChannel==NULL ) {

fprintf(stderr, "cannot allocate memory for LChannel in wav\_read\_fn\n");

fclose(fp);

return 0;

}

p\_wav->RChannel = (short \*) calloc(p\_wav->length, sizeof(short));

if( p\_wav->RChannel==NULL ) {

fprintf(stderr, "cannot allocate memory for RChannel in wav\_read\_fn\n");

fclose(fp);

return 0;

}

fseek(fp, 44, SEEK\_SET);

for(i=0;i<p\_wav->length;i++) {

fread(p\_wav->LChannel+i, sizeof(short), 1, fp);

fread(p\_wav->RChannel+i, sizeof(short), 1, fp);

}

fclose(fp);

return 1;

}

int wav\_save\_fn(char \*fn, wav \*p\_wav)

{

FILE \*fp = fopen(fn, "wb");

size\_t i;

if(fp==NULL) {

fprintf(stderr, "cannot save %s\n", fn);

return 0;

}

fwrite(p\_wav->header, sizeof(char), 44, fp);

for(i=0;i<p\_wav->length;i++) {

fwrite(p\_wav->LChannel+i, sizeof(short), 1, fp);

fwrite(p\_wav->RChannel+i, sizeof(short), 1, fp);

}

fclose(fp);

return 1;

}

int wav\_init(size\_t length, wav \*p\_wav)

{

p\_wav->length = length;

p\_wav->LChannel = (short \*) calloc(p\_wav->length, sizeof(short));

if( p\_wav->LChannel==NULL ) {

fprintf(stderr, "cannot allocate memory for LChannel in wav\_read\_fn\n");

return 0;

}

p\_wav->RChannel = (short \*) calloc(p\_wav->length, sizeof(short));

if( p\_wav->RChannel==NULL ) {

fprintf(stderr, "cannot allocate memory for RChannel in wav\_read\_fn\n");

return 0;

}

return 1;

}

void wav\_free(wav \*p\_wav)

{

free(p\_wav->LChannel);

free(p\_wav->RChannel);

}

/\* hamming: for n=0,1,2,...N, length of N+1 \*/

double hamming(int N, int n)

{

return 0.54 - 0.46 \* cosf(2\*PI\*((double)(n))/((double)N));

}

double band\_pass(int m, int n) {

float wl = 2\*PI\*FL/FS;

float wh = 2\*PI\*FH/FS;

if(n == m)

return 2.0\*(wh/PI - wl/PI);

else

return 2.0\*(sinf(wh\*((double)(n-m)))-sinf(wl\*((double)(n-m))))/PI/((double)(n-m)) \* hamming(2\*m+1, n);

}

double band\_stop(int m, int n){

double wl = 2\*PI\*FL/FS;

double wh = 2\*PI\*FH/FS;

if(n == m)

return 1.0 - (wh/PI - wl/PI);

else

return 1.0 - (sinf(wh\*((double)(n-m)))-sinf(wl\*((double)(n-m))))/PI/((double)(n-m)) \* hamming(2\*m+1, n);

}

void LogTransform(double \*X, int N) {

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

double re = log(abs(X[i]));

X[i] = re;

}

}

int main(int argc, char \*\*argv)

{

wav wavin;

wav wavout;

char fn\_in[1024] = {"blue\_giant\_fragment.wav"};

char fn\_out[1024] = {"out.wav"};

double h\_L[2\*M+1] = {0};

double h\_R[2\*M+1] = {0};

int n = 0;

double y = 0;

int k;

printf("start read wav\n");

// read wav

if( wav\_read\_fn(fn\_in, &wavin) == 0 ) {

fprintf(stderr, "cannot read wav file %s\n", fn\_in);

exit(1);

}

printf("start construct\n");

// construct low-pass filter

for(n=0;n<(2\*M+1);n++) {

h\_L[n] = band\_pass(M, n);

h\_R[n] = band\_stop(M, n);

}

/\*

for(n=0;n<(2\*M+1);n++) {

fprintf(stdout, "%.15f\n", h[n]);

}

\*/

// filtering (convolution)

if( wav\_init(wavin.length, &wavout)==0 ) {

exit(1);

}

printf("start change\n");

printf("wavin.length=%d\n",wavin.length);

for(n=0;n<wavin.length;n++) {

printf("n=%d\n",n);

y = 0;

for(k=0;k<(2\*M+1);k++) {

if( (n-k)>=0 )

y = y + h\_L[k] \* ((double)(wavin.LChannel[n-k]));

}

wavout.LChannel[n] = (short)(roundf(y));

y = 0;

for(k=0;k<(2\*M+1);k++) {

if( (n-k)>=0 )

y = y + h\_R[k] \* ((double)(wavin.RChannel[n-k]));

}

wavout.RChannel[n] = (short)(roundf(y));

}

printf("start change\n");

memcpy(wavout.header, wavin.header, 44);

printf("start save\n");

FILE\* fp = fopen(argv[2],"w");

for(int i=0;i<2\*M+1;i++){

fprintf(fp,"%e\n",h\_L[i]);

}

fclose(fp);

fp = fopen(argv[3],"w");

for(int i=0;i<2\*M+1;i++){

fprintf(fp,"%e\n",h\_R[i]);

}

fclose(fp);

double yl[1200], yr[1200];

int N = 20.06 \* FS;

// 复制出要分析的waveform片段

for(int i=0;i<1200;i++){

yl[i] = wavin.LChannel[i+N];

yr[i] = wavin.RChannel[i+N];

}

// FFT分析

LogTransform(yl, 1200);

fp = fopen(argv[4],"w");

for(int i=0;i<1200;i++){

fprintf(fp,"%e\n",yl[i]);

}

fclose(fp);

LogTransform(yr, 1200);

fp = fopen(argv[5],"w");

for(int i=0;i<1200;i++){

fprintf(fp,"%e\n",yr[i]);

}

fclose(fp);

// save wav

if( wav\_save\_fn(fn\_out, &wavout)==0) {

fprintf(stderr, "cannot save %s\n", fn\_out);

exit(1);

}

wav\_free(&wavin);

wav\_free(&wavout);

return 0;

}