AQUEST Corp.
http://www.a-quest.com/

AquesTalk2 Linux

Manual

1. 概要

本文書は、規則音声合成ライブラリ AquesTalk2 Linux をアプリケーションに組み込んで使用するためのプログラミングに関しての方法および注意点を示したものです。

AquesTalk2 は AquesTalk の後継として開発されました。合成音声の声質を規定するデータとして Phont を新しく定義し、これを差し替えることで様々な声の合成音声を生成することができます。また、この Phont は、別アプリケーションの PhontDesigner を用いれば、ユーザサイドで好みの声を作成することができます。

合成アルゴリズムは AquesTalk から大きく変更されました、ただし、入力の音声記号列の仕様および関数 IF は同じですので簡単に置き換えることができます（AquesTalk と AquesTalk2 の混在を考慮し関数名は異なります）。

AquesTalk Linux は Win 版の AquesTalk とは異なり、DA(サウンド出力)機能はありません。本ライブラリを用いて生成した音声波形データは、ファイルに出力、サウンドデバイスへ出力、またはネットワークを通じて転送するなど、アプリケーションの要求に応じた処理を別途実装する必要があります。

音声記号列の文字コードは、シフト JIS、EUC、UTF-8、UTF-16（LE)、ローマ字の各種を利用できます。それぞれ異なる名前の関数インターフェースが定義されています。

なお、本ライブラリをアプリケーションに組み込んで使用する際には、事前に当社ホームページ上の本ライブラリのライセンス規定を、ご確認ください。

2. 仕様

入力	かな表記音声記号列
出力	WAV フォーマット(8KHz サンプリング, 16bitPCM, モノラル)
声種	基本1種（Phont の差換えで変更可能）
ライブラリ形式	so 形式 共有ライブラリ
関数 I/F	C関数呼び出し __stdcall
OS	Linux x86-64(64bit 版)/ IA-32(32bit 版)
メモリ	コード：約50KB、ヒープメモリ：1スレッドあたり約10KB＋波形バッファ
マルチスレッド	対応
外部依存ライブラリ	libc.so

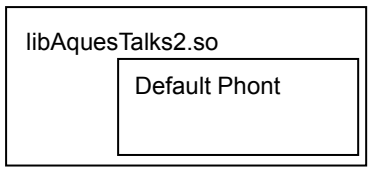
3. ライブラリ配置

AquesTalk2 Linux のライブラリは共有ライブラリとなっています。リンク時、および実行時に本ライブラリが必要になります。
以下に一例として、/usr/lib に本ライブラリを配置する方法を示します。これによりリンク時および実行時にライブラリにアクセスできるようになります。配置ディレクトリはお使いの環境に応じて変更してください(/usr/lib64 /usr/lib32 など)。また、バージョンによりライブラリのファイル名が実際と異なる場合があります。

以下を su 権限にて行います。

```
# cp libAquesTalk2.so.1.0 /usr/lib
# ln -sf /usr/lib/libAquesTalk2.so.1.0 /usr/lib/libAquesTalk2.so.1
# ln -sf /usr/lib/libAquesTalk2.so.1 /usr/lib/libAquesTalk2.so
# /sbin/ldconfig -n /usr/lib
```

4. Phont ファイル



パッケージの phont ディレクトリに、声種を規定する Phont ファイルがいくつか含まれています。なお、この中の aq_rm.phont は、ライブラリに内臓のデフォルト Phont と同じものです。
ライブラリ呼び出し時に Phontファイルデータを指定することによって、異なる声で合成が可能となります。
今後、新しい Phont ファイルを逐次公開する予定です。これらは、別途ダウンロードしてお使いください。
なお、Phont ファイルは Windows 用と Linux 用の違いはなく、相互に使用することができます。

5. コンパイル･リンク

本ライブラリを使用する場合、ライブラリを呼び出すソースコードに AquesTalk2 Linux のヘッダファイル(AquesTalk2.h)をインクルードしてコンパイルします。
リンク時には、 -lAquesTalk2 を指定してリンクしてください。
次に、サンプルプログラム SampleTalk.c をコンパイル・リンクする方法を示します。

```
$ g++ -o SampleTalk SampleTalk.c   -lAquesTalk2
```

g++の代わりに gcc でもリンクできますが、
undefined reference to `__gxx_personality_v0'　などのエラーが出る場合は、-lstdc++ を追加して指定してください。

6. 関数 API

AquesTalk2_Synthe　　AquesTalk2.h

説明	音声記号列から音声波形を生成します 生成した音声データは、使用後、呼び出し側で free()してください。
構文	unsigned char * **AquesTalk2_Synthe**(const char **koe*, int *iSpeed*, int **size, void *phontDat*)
引数	
koe	音声記号列
iSpeed	発話速度[%] 50-300 の間で指定 デフォルト:100 値を大きく設定するほど、速くなる
size	生成した音声データのサイズが返る[byte]（エラーの場合はエラーコードが返る）
phontDat	phont データの先頭アドレスを指定します。 この DLL のデフォルト Phont を用いるときは0を指定します。
戻り値	WAV フォーマットのデータ(内部で領域確保、解放は呼び出し側で AquesTalk2_FreeWave()で行う)の先頭アドレスを返す。エラー時は、NULL を返す。このとき size にエラーコードが設定される。

AquesTalk2_FreeWave　　AquesTalk2.h

説明	音声データの領域を開放
構文	void **AquesTalk2_FreeWave** (unsigned char *wav)

引数　なし

wav　WAV フォーマットのデータ（AquesTalk2_Synthe()で生成した音声データ）

戻り値　なし

AquesTalk2_Synthe_Euc　AquesTalk2.h

説明　音声記号列から音声波形を生成します

AquesTalk2_Synthe()の EUC 文字コード版

構文　unsigned char * **AquesTalk2_Synthe_Euc**(const char **koe*, int *iSpeed*, int * *size, void *phontDat*)

引数

koe　音声記号列(NULL 終端 EUC)

以下、 AquesTalk2_Synthe()と同じ

AquesTalk2_Synthe_Utf8　AquesTalk2.h

説明　音声記号列から音声波形を生成します

AquesTalk2_Synthe()の UTF-8 文字コード版

構文　unsigned char * **AquesTalk2_Synthe_Utf8**(const char **koe*, int *iSpeed*, int * *size, void *phontDat*)

引数

koe　音声記号列(NULL 終端 UTF-8 BOM は無し)

以下、 AquesTalk2_Synthe()と同じ

AquesTalk2_Synthe_Utf16　AquesTalk2.h

説明　音声記号列から音声波形を生成します

AquesTalk2_Synthe()の UTF-16 文字コード版

構文　unsigned char * **AquesTalk2_Synthe_Utf16**(const unsigned short *w*koe*, int *iSpeed*, int * *size, void *phontDat*)

引数

wkoe　音声記号列(NULL 終端 UTF-16 BOM 指定は任意　エンディアンは実行環境に依存)

以下、 AquesTalk2_Synthe()と同じ

AquesTalk2_Synthe_Roman　AquesTalk2.h

説明　音声記号列から音声波形を生成します

AquesTalk2_Synthe()のローマ字(7bitASCII)文字コード版

構文　unsigned char * **AquesTalk2_Synthe_Roman**(const char **koe*, int *iSpeed*, int * *size, void *phontDat*)

引数

koe　音声記号列(NULL 終端 ASCII　表記方法はホームページ上の音声記号列仕様を参照)

以下、 AquesTalk2_Synthe()と同じ

7. 音声データ形式

本ライブラリで生成する音声データは、次の形式となります。
AquesTalk2_Synthe()で生成する音声データには、先頭部に WAV ヘッダが付与されています。
ストレート PCM データが必要な場合は、別途ヘッダを除いて使用してください。
ほかのフォーマットをご希望の場合はカスタマイズを承りますので、お問い合わせください。

サンプリング周波数	8KHz
量子化 bit 数	16bit
チャンネル数	モノラル
エンコード	リニア PCM
フォーマット	WAV 形式

8. 音声記号列

音声記号列の書き方、仕様については、
http://www.a-quest.com/download/ の「音声記号列仕様」を参照ください。

9. サンプルコード

次に示すコードは、標準入力で音声記号列を指定し、この音声記号列から音声データを生成し、WAV ファイルとして出力する最も単純なプログラムです(パッケージに含まれている SampleTalk.c と同じ)。

SampleTalk.c

```
#include <stdio.h>
#include <AquesTalk2.h>     // AquestTalk クラスのヘッダ

int main(int ac, char **av)
{
    int  size;
    int  iret;
    char    str[1024];

    // 音声記号列を入力
    if(fgets(str, 1024-1, stdin)==0) return 0;

    // 音声合成
    unsigned char *wav = AquesTalk2_Synthe_Euc(str, 100, &size);
    if(wav==0){
        fprintf(stderr, "ERR:%d¥n",size);
        return -1;
    }

    // 音声データ(wav フォーマット)の出力
    fwrite(wav, 1, size, stdout);

    // 音声データバッファの開放
    AquesTalk2_FreeWave(wav);

    return 0;
}
```

ビルド方法

```
$ g++ -o SampleTalk SampleTalk.c   -lAquesTalk2
```

実行方法

```
$ echo "これわ、ごーせー/お'んせーです。" | ./SampleTalk > test.wav
```

音声記号列にシェルの特殊文字が含まれているので、ダブルクォーテーション(")で囲むなど、適当にエスケープする必要があります。

test.wav に音声データが生成できれば OK です。エラーの場合には、文字コードやライブラリの配置が正しく行われているか再確認してください。
本サンプルプログラムの文字コードは EUC になっています。文字コードを変更する場合は、AquesTalk2_Synthe_Euc() 部分を文字コードに応じて変更してください。

次に示すコードは、先のコードに外部の Phont ファイルを指定できるようにしたものです。
Phont ファイルをバイナリで読み込んで、先頭アドレスを合成時に指定しています。

```
#include <stdio.h>
#include <memory.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <AquesTalk2.h>     // AquestTalk クラスのヘッダ

void * file_load(const char * file, int * psize);

int main(int ac, char **av)
{
    int size;

    // Phont ファイルの読み込み
    int size;
    void *pPhont = file_load("¥¥aq_f1b.phont", &size); // Phont ファイルをここで指定
    if(pPhont ==0) return-2;

    //  メモリ上に音声データを生成
    unsigned char *wav = AquesTalk2_Synthe("こんにちわ。", 100, &size, pPhont);
    if(wav==0) {
        fprintf(stderr, "ERR %d", size); //  エラー時は size にエラーコードが返る
        return -1;
    }
    // Phont データの開放（音声合成が終わったら開放できる）
    free(pPhont);

    //    ルートディレクトリに生成した音声データを保存
    FILE *fp = fopen("¥¥ZZZ.wav", "wb");
    fwrite(wav, 1, size, fp);
    fclose(fp);

    // Synthe()で生成した音声データは、使用後に呼び出し側で解放する
    AquesTalk2_FreeWave (wav);

    return 0;
}

//  ファイルの読み込み
void * file_load(const char * file, int * psize)
{
    FILE    *fp;
    char    *data;
    struct stat  st;

    *psize = 0;

    if( stat(file, &st)!=0) return NULL;

    if((data=(char *)malloc(st.st_size))==NULL){
        fprintf(stderr,"can not alloc memory(file_load)¥n");
        return NULL;
    }
    if((fp=fopen(file,"rb"))==NULL) {
        free(data);
        perror(file);
        return NULL;
    }
    if(fread(data, 1, st.st_size, fp)<(unsigned)st.st_size) {
        fprintf(stderr,"can not read data (file_load)¥n");
        free(data);
```

```
            fclose(fp);
            return NULL;
        }
        fclose(fp);
        *psize = st.st_size;
        return data;
}
```

10. エラーコード表

関数が返すエラーコードの内容は、次の通りです。

値	内容
100	その他のエラー
101	メモリ不足
102	音声記号列に未定義の読み記号が指定された
103	韻律データの時間長がマイナスなっている
104	内部エラー(未定義の区切りコード検出)
105	音声記号列に未定義の読み記号が指定された
106	音声記号列のタグの指定が正しくない
107	タグの長さが制限を越えている(または[>]がみつからない)
108	タグ内の値の指定が正しくない
109	WAVE 再生ができない(サウンドドライバ関連の問題)
110	WAVE 再生ができない(サウンドドライバ関連の問題 非同期再生)
111	発声すべきデータがない
200	音声記号列が長すぎる
201	1つのフレーズ中の読み記号が多すぎる
202	音声記号列が長い(内部バッファオーバー1)
203	ヒープメモリ不足
204	音声記号列が長い(内部バッファオーバー1)
1000 - 1008	Phont データが正しくない

11. 履歴

日付	版	変更箇所	更新内容	更新者
2010/01/25	1.0	新規作成		N.Y