

AqKanji2Koe-A Linux マニュアル

株式会社 アクエスト
www.a-quest.com

概要

本文書は、音声合成ライブラリ **AqKanji2Koe-A 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の方法、注意点を示したものです。

AqKanji2Koe-A は、漢字かな混じり文のテキスト情報を **AquesTalk** 用のアクセント付きの音声記号列に変換するライブラリです。

このライブラリと音声合成ライブラリ **AquesTalk** を使えば、動的に変化する様々なテキストから、リアルタイムに音声メッセージを生成できるようになります。

なお、**AquesTalk pico** 用のローマ字の音声記号列を出力する関数も含まれています。

特長

- ・簡単に組み込み可能
 - テキスト文字列を入力すると音声記号列を返す、シンプルな API
- ・高速な変換処理
 - 約 20 万字/秒の高速な変換処理
- ・登録単語の追加が可能
 - ユーザ辞書ライブリを用いて、アプリで単語の追加や、編集、削除が可能
- ・高精度な読み・アクセント付与
 - 最大 38 万語の単語辞書とアクセントルールにより、正確な読みとアクセントを生成

なお、OS を持たないような小型組み込み機器用に、より小型軽量の **AqKanji2Koe-M** があります。こちらはパッケージ版は無く、すべて環境に応じてビルドして提供するカスタム製品となります。

本ライブラリを使ったプログラミングに先立って動作を確認される場合は、弊社サイトのオンラインデモや、Windows アプリ **AquesTalkPlayer** をダウンロードしてお試してください。漢字を含んだ文を音声記号列に変換や、ユーザ辞書を編集する機能を確認できます。

オンラインデモ

<https://www.a-quest.com/demo/>

AquesTalkPlayer

<https://www.a-quest.com/products/aquestalkplayer.html>

本ライブラリを使用するには開発ライセンスキーの設定が必要です。このライセンスキーを設定しない場合は評価版として動作し、以下の制限があります。

評価版の制限

「ナ行、マ行」は、すべて「ヌ」と出力されます

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**使用ライセンス**、配布には**頒布ライセンス**が必要です。ライセンスの種類や購入方法は、弊社サイトのライセンスのページを参照してください。

仕様

ライブラリ形式	so 形式 共有ライブラリ
対応 OS	Linux 64bit(x86_64) / 32bit(i386)
入力データ形式	漢字かな混じり文テキスト(UTF8,BOM 無し)
出力データ形式	かな/ローマ字 表記音声記号列(UTF8,BOM 無し)
関数 I/F	C 関数呼び出し
マルチスレッド	対応
ライブラリサイズ	約 200KByte
辞書サイズ	標準:約 10MB(約 38 万語)
処理速度	約 20 万文字/秒 (Windows XP, AthlonX2 2.7GHz 環境で計測)
ビルド環境	g++ 11.4.0
依存ライブラリ	libstdc++.so.6, libc.so.6, libgcc_s.so.1, libm.so.6

ビルド・実行

ライブラリ配置

AqKanji2Koe-A 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/lib64` に本ライブラリを配置・登録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/lib /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 libAqKanji2Koe..so.4.0 /usr/lib64
# cp libAqUsrDic..so.4.0 /usr/lib64
# ln -sf /usr/lib64/libAqKanji2Koe..so.4.0 /usr/lib64/libAqKanji2Koe..so.4
# ln -sf /usr/lib64/libAqKanji2Koe..so.4 /usr/lib64/libAqKanji2Koe.so
# ln -sf /usr/lib64/libAqUsrDic..so.4.0 /usr/lib64/libAqUsrDic..so.4
# ln -sf /usr/lib64/libAqUsrDic..so.4 /usr/lib64/libAqUsrDic.so
# /sbin/ldconfig -n /usr/lib64
```

アプリのビルド

プログラムのコンパイル時にはヘッダファイル(AqKanji2Koe.h)をインクルードします。必要に応じてインクルードファイルのパスを設定しておきます。リンク時には、libAqKanji2Koe.so ファイルをリンクするように-lAqKanji2Koe オプションを追加します。

パッケージ内の lib32 フォルダが 32bit 版、lib フォルダが 64bit 版となります。

依存ライブラリ

libAqKanji2Koe/libAqUsrDic は、内部では C++で記述しているため、アプリの実行時には C++と C のランタイムライブラリが必要となります。

ユーザ辞書ライブラリ libAqUsrDic

ユーザ辞書に単語の追加や削除、編集を行う場合には、別途、ユーザ辞書ライブラリ libAqUsrDic を使います。アプリに組み込む際は、libAqKanji2Koe と同様の方法で、AqUsrDic.h, libAqUsrDic.so を使用します。

関数 API

libAqKanji2Koe.so

AqKanji2Koe_Create

AqKanji2Koe.h

説明 言語処理モジュールのインスタンス生成と内部データの初期化
生成したインスタンスは、使用後に AqKanji2Koe_Release で解放すること。

構文 void * **AqKanji2Koe_Create** (const char *pathDic, int *pErr)

引数

pathDic 辞書のディレクトリを指定。通常、"<app dir>/aq_dic"。指定した内容は内部で保存される。
文字列最後の、/ の付与は任意

pErr エラー時にはエラーコードが入る 正常終了時は不定値

戻り値 インスタンスハンドル エラーの時は0が返る。このとき pErr にエラーコードが設定される。

AqKanji2Koe_Create_Ptr

AqKanji2Koe.h

説明 言語処理モジュールのインスタンス生成と内部データの初期化

生成したインスタンスは、使用後に `AqKanji2Koe_Release` で解放すること。
辞書データの先頭アドレスを指定するので、メモリマップドファイルでも使用可能。

構文 `void * AqKanji2Koe_Create_Ptr (const void *pSysDic, const void *pUserDic, int *pErr)`

引数

pSysDic システム辞書(通常 `aqdic.bin`) をメモリ上に読み込んだ先頭アドレスを指定

pUserDic ユーザ辞書(通常 `aq_user.dic`) をメモリ上に読み込んだ先頭アドレスを指定
ユーザ辞書を使用しない場合は `NULL` を指定する

pErr エラー時にはエラーコードが入る 正常終了時は不定値

戻り値 インスタンスハンドル エラーの時は0が返る。このとき *pErr* にエラーコードが設定される。

AqKanji2Koe_Release

`AqKanji2Koe.h`

説明 言語処理モジュールのインスタンスを開放

構文 `void AqKanji2Koe_Release (void * hAqKanji2Koe)`

引数

hAqKanji2Koe `AqKanji2Koe_Create()` / `AqKanji2Koe_Create_Ptr()`で返されたハンドルを指定

戻り値 なし

AqKanji2Koe_Convert

`AqKanji2Koe.h`

説明 漢字かな混じりのテキストをカナ音声記号列に変換

構文 `int AqKanji2Koe_Convert (void * hAqKanji2Koe, const char *kanji, char *koe, int nBufKoe)`

引数

hAqKanji2Koe `AqKanji2Koe_Create()` / `AqKanji2Koe_Create_Ptr()`で返されたハンドルを指定

kanji 入力漢字かな混じり文テキスト文字列 (UTF8, NULL 終端、BOM 無し)

koe 出力バッファ。かな音声記号列文字列が返る (UTF8, NULL 終端、BOM 無し)

nBufKoe *koe* バッファのサイズ[byte] 256 以上を指定。バッファ変換後の音声記号列がこのサイズを超えるとエラーを返すので、入力テキストの2倍以上のサイズを指定することを推薦。

戻り値 0:正常終了 それ以外:エラーコード

説明	漢字かな混じりのテキストをローマ字の音声記号列に変換
構文	int AqKanji2Koe_ConvRoman (void * <i>hAqKanji2Koe</i> , const char * <i>kanji</i> , char * <i>koe</i> , int <i>nBufKoe</i>)
引数	
<i>hAqKanji2Koe</i>	AqKanji2Koe_Create() / AqKanji2Koe_Create_Ptr() で返されたハンドルを指定
<i>kanji</i>	入力漢字かな混じり文テキスト文字列 (UTF8, NULL 終端、BOM 無し)
<i>koe</i>	出力バッファ。ローマ字音声記号列文字列が返る (UTF8, NULL 終端、BOM 無し)
<i>nBufKoe</i>	<i>koe</i> バッファのサイズ [byte] 256 以上を指定。バッファ変換後の音声記号列がこのサイズを超えるとエラーを返すので、入力テキストの2倍以上のサイズを指定することを推薦。
戻り値	0:正常終了 それ以外:エラーコード

説明	開発ライセンスキーを設定。アプリ起動後、他の関数を呼び出す前に呼び出すことで、以降、製品版とし動作し、評価版の制限がなくなる。
構文	int AqKanji2Koe_SetDevKey (const char * <i>key</i>)
引数	
<i>key</i>	開発ライセンスキー文字列 (半角英数)
戻り値	ライセンスキーが正しければ 0、正しくなければ 1 が返る。 不正なキーでも 0 を返す場合がある。このとき制限は解除されない。

libAqUsrDic.so (ユーザ辞書ライブラリ)

AqUsrDic ライブラリは、個々の単語の登録を行うインターフェースはありません。アプリケーションでユーザ辞書を変更するときは、CSV 形式の単語リストファイルを介して行います。

AqUsrDic_Import

AqUsrDic.h

説明 CSV 形式の単語リストからユーザ辞書(aq_usr.dic)を生成
生成する aq_usr.dic と同じディレクトリに、システム辞書(aqdic.bin)が必要。
aq_usr.dic がすでにある場合は上書きする(Appendしない)。
【注意】ユーザ辞書は生成時のシステム辞書に依存するため、異なるシステム辞書で生成したユーザ辞書は互換性が無い。

構文 int **AqUsrDic_Import**(const char * *pathUserDic*, const char * *pathDicCsv*)

引数

pathUserDic 出力するユーザ辞書(aq_user.dic)ファイルを指定

pathDicCsv CSV 単語リストファイルファイルを指定(入力データ)

戻り値 0:正常終了 それ以外:エラーコード(詳細は AqUsrDic_GetLastError()で取得)

AqUsrDic_Export

AqUsrDic.h

説明 ユーザ辞書(aq_usr.dic)から CSV 形式の単語リストを生成
aq_usr.dic と同じディレクトリに、システム辞書(aqdic.bin)が必要。

構文 int **AqUsrDic_Export**(const char * *pathUserDic*, const char * *pathDicCsv*)

引数

pathUserDic ユーザ辞書(aq_user.dic)ファイルを指定

pathDicCsv 出力する CSV 単語リストファイルファイルを指定

戻り値 0:正常終了 それ以外:エラーコード(詳細は AqUsrDic_GetLastError()で取得)

AqUsrDic_Check

AqUsrDic.h

説明 単語表記、読みの書式や品詞コードが正しいかチェックする

構文 int **AqUsrDic_Check**(const char * *surface*, const char * *yomi*, int *posCode*)

引数

<i>surface</i>	単語の表記の文字列を指定(UTF8)
<i>yomi</i>	単語の読み(アクセント付き音声記号列)の文字列を指定(UTF8)
<i>posCode</i>	単語の品詞コード(下記参照)を指定
戻り値	0:チェック OK それ以外:NG(詳細は AqUsrDic_GetLastError()で取得)

AqUsrDic_GetLastError

AqUsrDic.h

説明	最後のエラーの詳細メッセージを返す
構文	const char * AqUsrDic_GetLastError()
引数	なし
戻り値	エラーメッセージ(UTF8, NULL 終端)

エラーコード表

libAqKanji2Koe ライブラリの関数が返すエラーコードの内容は、次の通りです。
libAqUsrDic ライブラリに関しては、AqUsrDic_GetLastError()関数で内容を取得してください。

値	内容
100	その他のエラー
101	関数呼び出し時の引数が NULL になっている。
104	初期化されていない(初期化ルーチンが呼ばれていない)
105	入力テキストが長すぎる
106	システム辞書データが指定されていない
107	変換できない文字コードが含まれている
200 番台	システム辞書(aqdic.bin)が不正
300 番台	ユーザ辞書(aq_user.dic)が不正

CSV 単語リストファイル

ユーザ辞書の編集で使用する CSV 単語リストファイルの例を下に示します。
1 列目: 単語の表記を記述します。文字コードは UTF8 です。すべて全角で記述します。空白は指定できません。

2列目：単語の読みを音声記号列で指定します(UTF8)。アクセントも ' 記号で指定できます。詳細は音声記号列仕様書の読み記号とアクセント記号の項を参照ください。区切記号は指定できません。
3列目：単語の品詞コード(下記参照)を半角数値で指定します。

CSV 単語リストファイルの例

```
大江戸線,オーエドセン,5  
GDGD,ガ'ダダ,1  
天王洲,テンノーズ,7  
Jアラート,ジェイアラート,5  
就活,シューカツ,1  
魔理沙,マ'リサ,4  
どや顔,ドヤガオ,1  
アラフォー,アラフォー,0
```

品詞コード一覧

0	名詞
1	名詞(サ変)
2	人名
3	人名(姓)
4	人名(名)
5	固有名詞
6	固有名詞(組織)
7	固有名詞(地域)
8	固有名詞(国)
9	代名詞
10	代名詞(縮約)
11	名詞(副詞可能)
12	名詞(接続詞的)
13	名詞(形容動詞語幹)
14	名詞(ナイ形容詞語幹)
15	形容詞
16	副詞
17	副詞(助詞類接続)
18	接頭詞(名詞接続)
19	接頭詞(動詞接続)
20	接頭詞(数接続)
21	接頭詞(形容詞接続)
22	接続詞
23	連体詞
24	記号
25	記号(アルファベット)
26	感動詞
27	間投詞

サンプルプログラム

コード説明

次に示すコードは、日本語のテキストを標準入力から読み込んで、音声記号列を標準出力に出力するだけの単純なプログラムです(SDK パッケージ内の/samples/Kanji2KoeCmd.cpp と基本的に同じものです)。

基本的な処理の流れは、最初に言語処理部のインスタンスを生成し (25 行目)、1 行毎にテキストを読み込んで音声記号列に変換して出力しています。最後の行を処理したら AqKanji2Koe_Release() でインスタンスを開放しています。

このプログラムで出力した音声記号列を AquesTalk に与えることで音声データに変換することができます。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <locale.h>    // setlocale()
5  #include <AqKanji2Koe.h>
6
7  #define  NSTR 4096
8
9  char * GetPathDic(const char *pathModule);
10
11 int main(int ac, char **av)
12 {
13     int iret;
14     char kanji[NSTR];
15     char koe[NSTR];
16     void *hAqKanji2Koe;
17
18     // iret = AqKanji2Koe_SetDevKey("XXX-XXX-XXX");
19     if(ac==1){
20         char *pPathDic  = GetPathDic(av[0]);
21         hAqKanji2Koe = AqKanji2Koe_Create(pPathDic, &iret);
22         free(pPathDic);
23     }
24     else {
25         hAqKanji2Koe = AqKanji2Koe_Create(av[1], &iret);
26     }
27
28     if(hAqKanji2Koe==0){
29         fprintf(stderr, "ERR: can not initialize Dictionary(%d)\n", iret);
30         fprintf(stderr, "USAGE: $ Kanji2KoeCmd (dic_dir) < in.txt > out.koe\n");
31         return iret;
32     }
33
34     int i;
35     for(i=0; ; i++){
36         if(fgets(kanji, NSTR, stdin)==0) break;
37         iret = AqKanji2Koe_Convert(hAqKanji2Koe, kanji, koe, NSTR);
38         if(iret!=0) {
39             fprintf(stderr, "ERR: AqKanji2Koe_Convert()=%d\n", iret);
40             break;
41         }

```

```
42     fprintf(stdout, "%s\n", koe);
43     }
44
45     AqKanji2Koe_Release(hAqKanji2Koe);
46     return 0;
47 }
```

`AqKanji2Koe_Create()`でインスタンスを生成する場合、辞書データは内部メモリに一旦すべて読み込まれます。

一方、`AqKanji2Koe_Create_Ptr()`では、呼び出し側で辞書データを読み込みます。例えば、サーバ利用などで複数の言語処理プロセスを起動する場合は、メモリマップトファイルを使って複数のプロセスで辞書データのメモリを共有することができます。

他の言語（Python、PHP、Ruby など）での使用方法はここでは示しませんが、呼び出して使用することが出来ると思います。

ビルド

SDK パッケージにはサンプルアプリのソースコードが2つ含まれています。`Kanji2KoeCmd.cpp`は、上記サンプルと同じものです。以下にコンパイル、リンク方法を示します。

```
$ g++ -I./lib -o Kanji2KoeCmd samples/Kanji2KoeCmd.cpp -lAqKanji2Koe
```

`-I./lib` で、ヘッダファイル `AqKanji2Koe.h` のパスを指定しています。
`-lAqKanji2Koe` で、`libAqKanji2Koe.so` をリンクしています。
上記コマンドで、実行モジュール、`Kanji2KoeCmd` が出来るはずですが。

実行

実行時には、辞書データを指定された場所に配置しておく必要があります。`Kanji2KoeCmd` サンプルプログラムでは実行モジュールと同じディレクトリに辞書ファイルのフォルダ(`aq_dic`)を配置してください。このときのファイルの配置は以下のとおりです。

```
| - Kanji2KoeCmd
| - aq_dic/
|   | - aqdic.bin
|   | - aq_user.dic (ユーザ辞書は必要に応じて)
```

端末から `./Kanji2KoeCmd` を実行すると、入力待ち状態になります。ここで適当な文を入力すると、音声記号列が出力されます。

```
$ echo 音声合成テスト | ./Kanji2KoeCmd
オンセーゴーセーテスト。
$
```

アプリ開発ガイドライン

アプリケーションの開発(評価での使用を除く)は、以下のガイドラインに従ってください。

ライセンスキー

本ライブラリの動作は、開発ライセンスキーに依存します。このキーは、各ライセンス購入時に発行されるライセンス証に記載されています。

`AquesTalk.setDevKey ()` をアプリケーションの起動初期に一度呼び出します。引数には開発ライセンスキーを指定します。これにより製品版として動作し、評価版の制限がなくなります。

CREDITS

本ライブラリは、BSD ライセンスに基づいてライセンスされている下記のオープンソースソフトウェアを使用しています。このライセンスの表示は SDK 付属の **CREDITS** ファイルを参照ください。また、本ライブラリを含んだアプリを配布する場合は、**CREDITS** ファイルまたはその内容が含まれるようにしてください。

- MARISA: Matching Algorithm with Recursively Implemented StorAge
- NAIST Japanese Dictionary：形態素解析用辞書

文書履歴

日付	版	更新内容
2011/01/17	1.0	新規作成
2013/06/26	2.0	Ver.2 用書き換え
2017/11/14	3.0	Ver.3 用書き換え
2019/03/04	4.0	Ver.4 用書き換え
2022/12/07	4.1.3	ローマ字音声記号列の関数追加