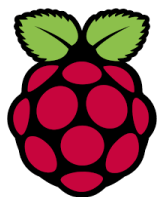


WLOラズパイ倶楽部+α

音声認識エンジンJuliusで ボイスコマンドに挑戦

2019年8月22日




Julius

<http://git.io/wlopi>

・ 前回(2019/7/11)以降の新情報

Raspbian buster 2019-07-10リリース




Raspbian Buster with desktop and recommended software
Image with desktop and recommended software based on Debian Buster

Version: July 2019
Release date: 2019-07-10
Kernel version: 4.19
Size: 1945 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256:
2bd0613ec8739b6fa4274ea186ec859046f79e6aee4b8c6af0acb6d88f3f533a




Raspbian Buster with desktop
Image with desktop based on Debian Buster

Version: July 2019
Release date: 2019-07-10
Kernel version: 4.19
Size: 1149 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256:
6a1a5f20329e580d5161a0255b3d4163db6f56c3997e1c3b36bdd51140bd768e



Raspbian Buster Lite
Minimal image based on Debian Buster

Version: July 2019
Release date: 2019-07-10
Kernel version: 4.19
Size: 426 MB

[Release notes](#)

[Download Torrent](#) [Download ZIP](#)

SHA-256:
9e5cf24ce483bb96e7736ea75ca422e3560e7b455eee63dd28f66fa1825db70e

前回のインストールガイド時に発生した問題も

改善していますので、新規インストール時には2019-07-10をお使いください。

更新情報

http://downloads.raspberrypi.org/raspbian/release_notes.txt

(日本語訳)

<https://qiita.com/utaani/items/2d4e641eba3d2e46d9f3>

<http://git.io/wlopi>

■ Raspberry PiにJuliusを導入して、音声認識システムを体感する

- ・ USB オーディオ入力のセットアップ
- ・ juliusツールとディクテーションキットの導入
- ・ 認識率向上のための最適化方法を学ぶ
 - ・ 単語辞書による認識
 - ・ 文法ファイルによる認識
- ・ 認識結果の活用

・ Julius: 汎用大語彙連続音声認識エンジン

Juliusは、汎用大語彙連続音声認識エンジンで、京都大学、名古屋工業大学などの開発者が開発、公開しているオープンソースソフトウェアです。音響モデル、単語辞書、言語モデルを準備することで、高性能な日本語音声認識システムを作ることができます。



Julius

now on **GitHub**

GitHub

- GitHub site
- ダウンロード
- Julius最新版
- Julius音声認識パッケージ
- 文法認識キット
- 音素セグメンテーションキット
- 言語モデル・音響モデル
- ドキュメント
- The Juliusbook
- チュートリアル・解説
- インストール
- アプリケーション開発
- マニュアル・ソース資料
- ユーザフォーラム

What's Julius?

Julius は、音声認識システムの開発・研究のためのオープンソースの高性能な汎用大語彙連続音声認識エンジンです。数万語彙の連続音声認識を一般のPCやスマートフォン上でほぼ実時間で実行できる軽量さとコンパクトさを持っています。

言語モデルとして単語N-gram、記述文法、ならびに単語辞書を用いることができます。また音響モデルとしてトライフォンのGMM-HMMおよびDNN-HMMを用いたリアルタイム認識を行うことができます。DNN-HMMの出力計算にnumpyを用いた外部モジュールを利用することも可能です。複数のモデルや複数の文法を並列で用いた同時認識も行うことができます。

Juliusの最大の特徴はその可塑性にあります。単語辞書や言語モデル・音響モデルなどの音声認識の各モジュールを組み替えることで、小語彙の音声対話システムからディクテーションまで様々な幅広い用途に应用できます。

Julius はオープンソースソフトウェアです。プログラムはC言語で書かれており、さまざまなプラットフォームへの移植や改造が容易です。ライセンスはオープンライセンスで、商用利用への制限もありません。

Julius の研究・開発に関わっている主な機関は以下の通りです。

Copyright (c) 1991-2019 京都大学 河原研究室

Copyright (c) 1997-2000 情報処理振興事業協会 (IPA)

Copyright (c) 2000-2005 奈良先端科学技術大学院大学 鹿野研究室

Copyright (c) 2005-2019 名古屋工業大学 Julius開発チーム

<https://julius.osdn.jp/>

Julius: Open-Source Large Vocabulary Continuous Speech Recognition Engine

DOI 10.5281/zenodo.2530396

Copyright (c) 1991-2019 Kawahara Lab., Kyoto University

Copyright (c) 2005-2019 Julius project team, Lee Lab., Nagoya Institute of Technology

Copyright (c) 1997-2000 Information-technology Promotion Agency, Japan

Copyright (c) 2000-2005 Shikano Lab., Nara Institute of Science and Technology

About Julius

"Julius" is a high-performance, small-footprint large vocabulary continuous speech recognition (LVCSR) decoder software for speech-related researchers and developers. Based on word N-gram and context-dependent HMM, it can perform real-time decoding on various computers and devices from micro-computer to cloud server. The algorithm is based on 2-pass tree-trellis search, which fully incorporates major decoding techniques such as tree-organized lexicon, 1-best / word-pair context approximation, rank/score pruning, N-gram factoring, cross-word context dependency handling, enveloped beam search, Gaussian pruning, Gaussian selection, etc. Besides search efficiency, it is also modularized to be independent from model structures, and wide variety of HMM structures are supported such as shared-state triphones and tied-mixture models, with any number of mixtures, states, or phone sets. It also can run multi-instance recognition, running dictation, grammar-based recognition or isolated word recognition simultaneously in a single thread. Standard formats are adopted for the models to cope with other speech / language modeling toolkit such as HTK, SRILM, etc. Recent version also supports Deep Neural Network (DNN) based real-time decoding.

The main platform is Linux and other Unix-based system, as well as Windows, Mac, Androids and other platforms.

Julius has been developed as a research software for Japanese LVCSR since 1997, and the work was continued under IPA Japanese dictation toolkit project (1997-2000), Continuous Speech Recognition Consortium, Japan (CSRC) (2000-2003) and Interactive Speech Technology Consortium (ISTC).

The main developer / maintainer is Akinobu Lee (ri@nitech.ac.jp).

<https://github.com/julius-speech/julius>

<http://git.io/wlopi>



・音響モデル-単語辞書/文法モデルで認識

音響モデル：トライフォンGMM-HMMおよびDNN-HMMを用いたリアルタイム認識

トライフォン：音素の前後のつながりも考慮したモデル

GMM-HMM：Gaussian-Mixed-Model Hidden Markov Model

混合ガウスモデル-隠れマルコフモデル

DNN-HMM：Deep-Neural-Network Hidden Markov Model

深層ニューラルネットワーク-隠れマルコフモデル

言語モデル：単語N-gram/記述文法/単語辞書

単語N-gram：辞書を使わず形態素解析による単語切り出し

記述文法：決められた文法による解析（「xxをn個ください」のような文法を定義）

単語辞書：単語辞書に当てはめて解析

- ・ **音響モデル/単語辞書/文法モデルで認識**
モデルと辞書をパッケージして配布している。

Julius音声認識パッケージ

Julius を動かしてみるために必要なキットです。日本語のディクテーション（自動口述筆記）に必要な最小限のモデルおよび Julius の実行バイナリが含まれていますので、これだけでとりあえず Julius を動かしてみることができます。現在、ディクテーションキット、話し言葉モデルキット、講演音声モデルキットの3つを公開しています。

ディクテーションキット (dictation-kit)

- 一般的なモデル
- JNASと『日本語話し言葉コーパス』模擬講演データによるDNN-HMM音響モデルとGMM-HMM音響モデルの2つを用意
- 国立国語研究所のBCCWJによる汎用的な言語モデル

478MB

話し言葉モデルキット (ssr-kit)

- 話し言葉認識を目的としたモデル
- JNASと『日本語話し言葉コーパス』模擬講演データによるDNN-HMM音響モデル
- 『日本語話し言葉コーパス』の模擬講演データと学会データによる言語モデル

306MB

講演音声モデルキット (lsr-kit)

- 大きな部屋等での講演を対象としたモデル
- 『日本語話し言葉コーパス』の学会データによるDNN-HMM音響モデル
- 『日本語話し言葉コーパス』の模擬講演データと学会データによる言語モデル

305MB

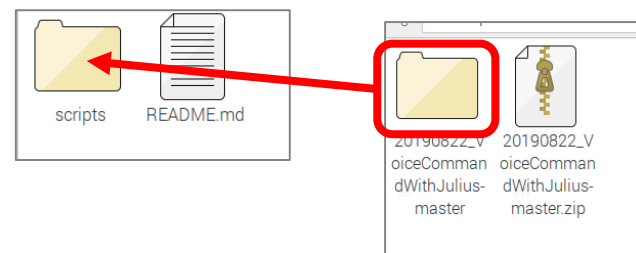
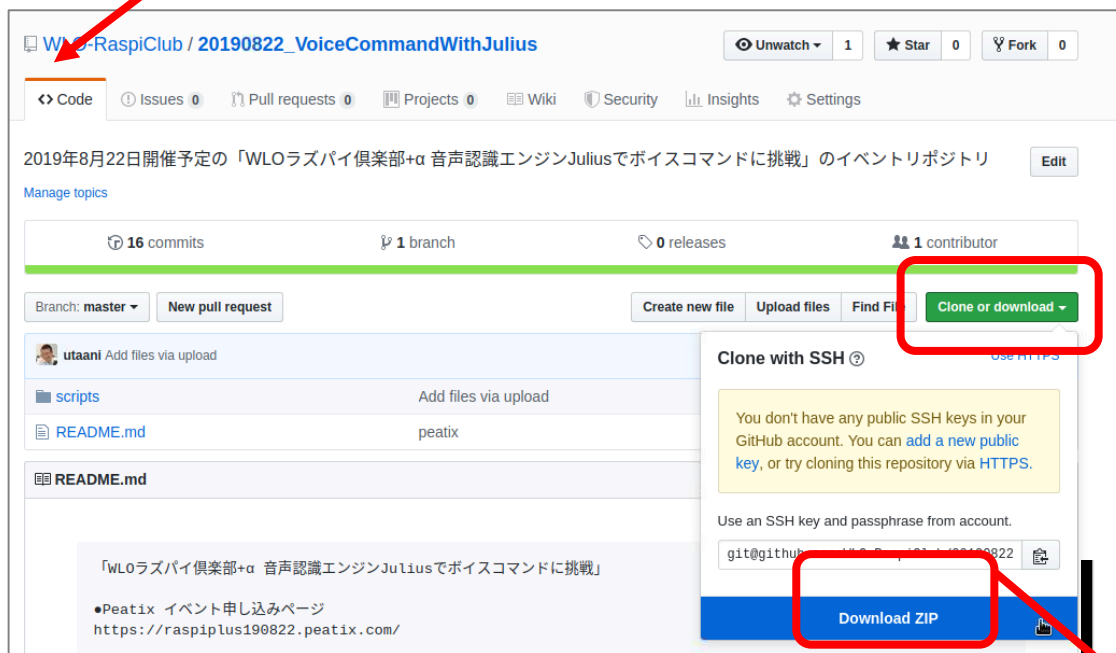
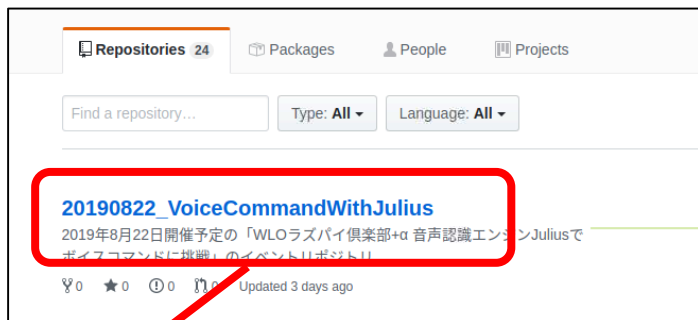
動作環境は Windows/Linux/MacOSX です。Juliusの実行バイナリを入れ替えば他のOSでも動きます。ssr-kit, lsr-kit の辞書登録ツールと音量チェックツール起動スクリプト adintool-gui.bat は Windows のみサポートしています。

<https://julius.osdn.jp/index.php?q=dictation-kit.html>

<http://git.io/wlopi>

■手順やファイルをgithubリポジトリに置いていきます

<https://github.com/WLO-RaspiClub/>
<https://git.io/wlopi>



■ Juliusの導入

- Raspbianに必要なパッケージを導入します

```
$ sudo apt install build-essential zlib1g-dev libasound2-dev flex
```

- 導入ディレクトリを /home/pi/julius に作成します

```
$ mkdir /home/pi/julius  
$ cd /home/pi/julius
```

- githubからjuliusをダウンロードします

```
$ git clone https://github.com/julius-speech/julius.git
```

- juliusのビルド&インストール /home/pi/julius/julius に保存されます

```
$ cd julius  
$ ./configure --with-mictype=alsa --enable-pthread  
<中略> 2分くらい  
  
$ make  
<中略> 3分くらい  
  
$ sudo make install
```

Juliusの導入確認

```
pi@raspberrypi:~ $ julius
Julius rev.4.5 - based on
JuliusLib rev.4.5 (fast) built for armv7l-unknown-linux-gnueabi

Copyright (c) 1991-2019 Kawahara Lab., Kyoto University
Copyright (c) 1997-2000 Information-technology Promotion Agency, Japan
Copyright (c) 2000-2005 Shikano Lab., Nara Institute of Science and Technology
Copyright (c) 2005-2019 Julius project team, Nagoya Institute of Technology

Try '-setting' for built-in engine configuration.
Try '-help' for run time options.
```

(参考) コマンドライン引数

https://julius.osdn.jp/juliusbook/ja/desc_options.html

コマンドライン引数として直接指定してもいいが、たくさん指定をすると長くなるので、ファイルに記述して
-C xxxx.jconf
で指定してもよい

付録 B. オプション一覧	
前のページ	次のページ
付録 B. オプション一覧	
目次	
アプリケーション	
全体オプション	
オーディオ入力	
レベルと零交差による入力検知	
入力棄却	
GMM / GMM-VAD	
デコーディング	
その他	
インスタンス宣言	
言語モデル (-LM)	
N-gram	
記述文法	
単語辞書 (孤立単語認識)	
ユーザ定義LM	
その他	
音響モデル・特徴量抽出 (-AM) (-AM_GMM)	
音響モデル・HMM	
特徴量抽出	
正規化処理	
フロントエンド処理	
その他	
認識処理・探索 (-SR)	
第1パスパラメータ	
第2パスパラメータ	
ショートポーズセグメンテーション / デコーダVAD	
単語ラティス / confusion network 出力	
複数文法認識	
Forced alignment	
その他	
Julius および JuliusLib コアエンジンの設定 (動作選択, 設定, モデル指定, パラメータ変更など) は、すべてここで説明する「オプション」で指定する。Julius に対しては、これらのオプションをコマンドライン引数として直接指定するか、あるいはテキストファイル内に記述したものを「-c」につづけて指定する。このオプションを記述したテキストファイルは「jconf 設定ファイル」と呼ばれる。	

```
pi@raspberrypi:~/dictation-kit $ cat dictation-kit-4
dictation-kit GMM-HMM configuration

# GMM-HMM
-h model/phone_m/jnas-tri-3k16-gid.binhmm
-hlist model/phone_m/logicalTri
-hlist model/phone_m/logicalTri-3k16-gid.bin

# language model weights and word insertion penalties
-lmp 10 0
-lmp2 10 0

# choose microphone input
-input mic

## EOF
```

■ USBマイク or オーディオインターフェースを接続する

- lsusb コマンドで接続を確認

```
pi@raspberrypi:~ $ lsusb
Bus 001 Device 005: ID 8086:0808 Intel Corp.
Bus 001 Device 004: ID 04f2:0918 Chicony Electronics Co., Ltd
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp. SMC95
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

- arecord -l で音声入力デバイス認識を確認

```
pi@raspberrypi:~ $ arecord -l
**** ハードウェアデバイス CAPTURE のリスト ****
カード 1: Device [USB PnP Sound Device], デバイス 0: USB Audio [USB Audio]
  サブデバイス: 1/1
  サブデバイス #0: subdevice #0
```

カード番号（上記だと1）とデバイス番号（上記だと0）を確認しておく
※複数の入力デバイスを持つUSBインターフェースの場合は、
とりあえず一番若い番号を試す

■環境変数によるデバイスの設定

- ALSADEV 環境変数を設定する

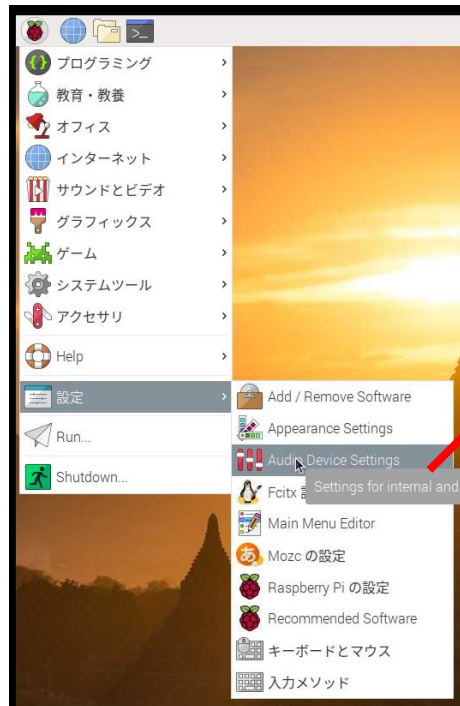
カード番号1 デバイス番号0 の場合、 `export ALSADEV="plughw:1,0"`

```
pi@raspberrypi:~ $ arecord -l
**** ハードウェアデバイス CAPTURE のリスト ****
カード 1: Device [USB PnP Sound Device], デバイス 0: USB Audio [USB Audio]
  サブデバイス: 1/1
  サブデバイス #0: subdevice #0
pi@raspberrypi:~ $ export ALSADEV="plughw:1,0"
```

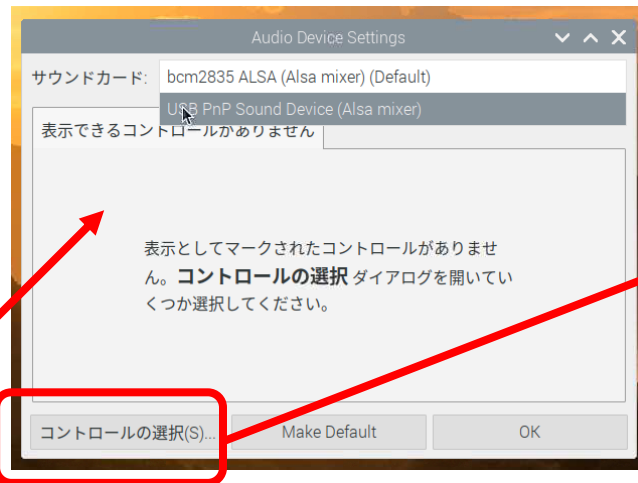
- ALSADEV 環境変数はログイン毎、仮想端末(LXTerminal)毎に設定が必要
自動で設定するには、`.bashrc`に記述する

■ USBマイク or オーディオインターフェースを接続する(承前)

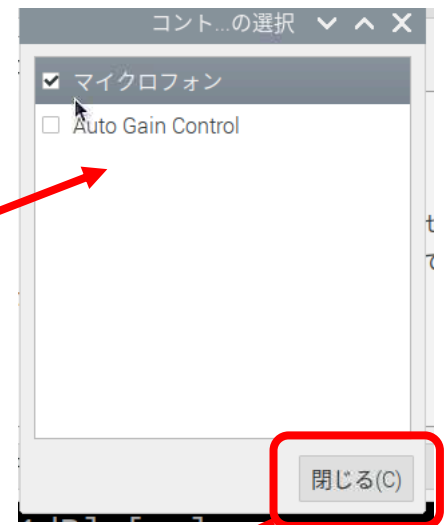
・ 録音ボリュームの設定 (GUIでの設定)



サウンドカードの切換え(USB PnP Sound Device)



マイクロフォンをチェック

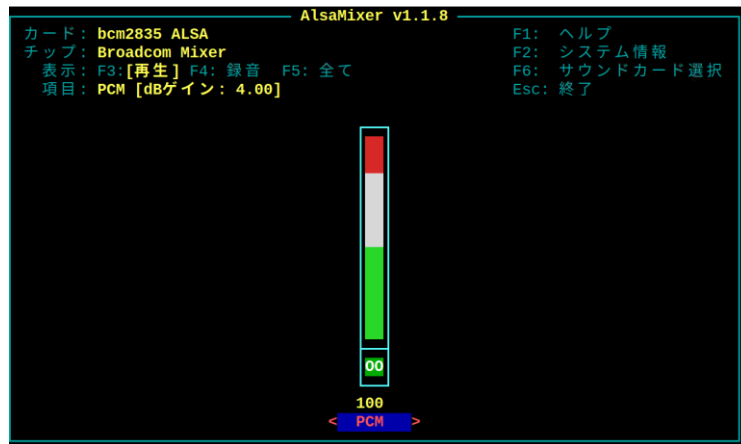


ボリュームを最大にして
OKで閉じる

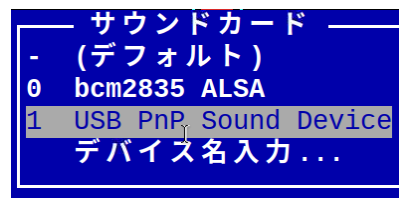


■ USBマイク or オーディオインターフェースを接続する(承前)

- ・ 録音ボリュームの設定 (CLIの設定: alsamixer コマンド)

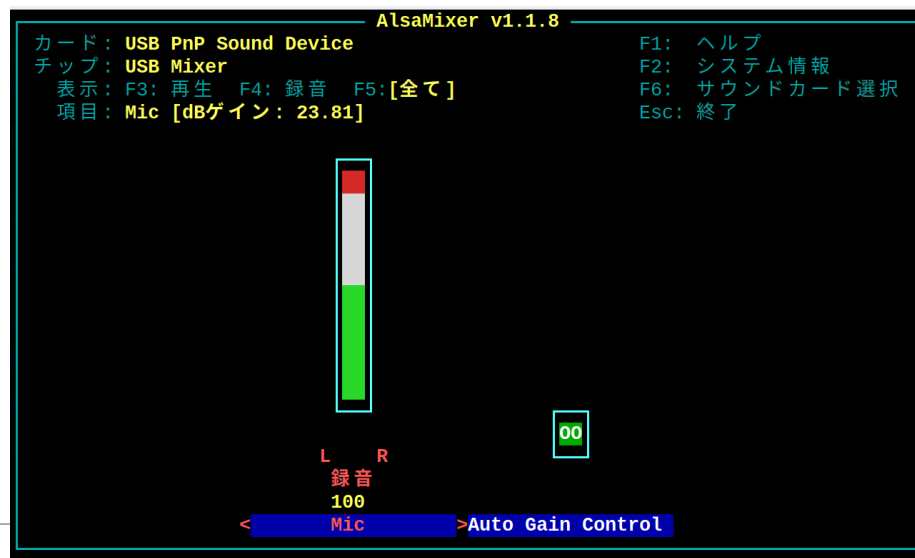


- ・ サウンドカード選択(F6)
→USB PnP Sound Deviceを選択(Enter)



- ・ 録音デバイス選択(F4)
→カーソルキーで100%にする

- ・ ESCで終了



■ オーディオのテスト

- ・ adintoolを使った確認

```
$ adintool -in mic -out file -filename test
```

<中略>

```
-----
STAT: ##### initialize input device
[start recording]
Stat: adin_alsa: device name from ALSADEV: "plughw:1,0"
Stat: capture audio at 16000Hz
Stat: adin_alsa: latency set to 32 msec (chunk = 512 bytes)
Stat: "plughw:1,0": Device [USB PnP Sound Device] device USB Audio [USB Audio] subdevice
#0
STAT: AD-in thread created
[test.0000.wav].....[segmented]
test.0000.wav: 20160 samples (1.26 sec.) [105280 ( 6.58s) - 125440 ( 7.84s)]
[test.0001.wav].....[segmented]
test.0001.wav: 17088 samples (1.07 sec.) [162112 (10.13s) - 179200 (11.20s)]
[test.0002.wav].....[segmented]
test.0002.wav: 14528 samples (0.91 sec.) [209728 (13.11s) - 224256 (14.02s)]
[test.0003.wav].....[segmented]
test.0003.wav: 15552 samples (0.97 sec.) [257856 (16.12s) - 273408 (17.09s)]
<<< please speak >>>^C[Interrupt]pi@raspberrypi:~ $
```

↑ Ctrl-Cで中断

センテンスごとに、分割されてファイルが作成される

```
pi@raspberrypi:~ $ ls *.wav
test.0000.wav  test.0001.wav  test.0002.wav  test.0003.wav
pi@raspberrypi:~ $
```

■ダウンロード

- ・ブラウザで <https://julius.osdn.jp/index.php?q=dictation-kit.html>

ダウンロード

以下から直接 zip 形式でダウンロードできます。最新版は 4.5 です。

ディクテーションキット (dictation-kit)

- ・ [ディクテーションキット version 4.5 \(https://osdn.net/dl/julius/dictation-kit-4.5.zip\)](https://osdn.net/dl/julius/dictation-kit-4.5.zip)
 - [version 4.4](#)
 - [version 4.3.1](#)

話し言葉モデルキット (ssr-kit)

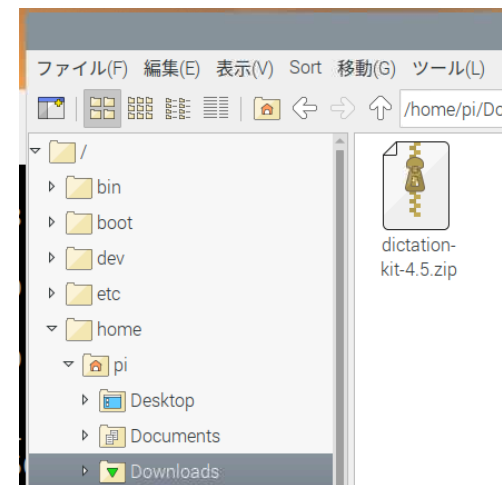
- ・ [話し言葉モデルキット version 4.5 \(https://osdn.net/dl/julius/ssr-kit-v4.5.zip\)](https://osdn.net/dl/julius/ssr-kit-v4.5.zip)
 - [version 4.4.2.1a](#)

講演音声モデルキット (lsr-kit)

- ・ [講演音声モデルキット version 4.5 \(https://osdn.net/dl/julius/lsr-kit-v4.5.zip\)](https://osdn.net/dl/julius/lsr-kit-v4.5.zip)
 - [version 4.4.2.1a](#)

なおディクテーションキットは[GitHub のディクテーションキットのページ](#)では最新の開発版を公開しています。git-lfs (Git Large File Storage) を使っていますので、clone する前に必ず git-lfs をインストール・設定してから clone して下さい。

478MBあるので、
会場では
USBメモリで
お配りします。
(Downloadsに
コピーしてください)



```
$ mv /home/pi/Downloads/dictation-kit-4.5.zip /home/pi/julius
```

- ・ コマンドラインからダウンロードする場合

```
$ cd /home/pi/Downloads  
$ wget https://osdn.net/dl/julius/dictation-kit-4.5.zip  
$ mv /home/pi/Downloads/dictation-kit-4.5.zip /home/pi/julius
```

<http://git.io/wlopi>

■ 導入

```
$ cd /home/pi/julius  
$ unzip dictation-kit-4.5.zip  
$ cd dictation-kit-4.5
```

■ ディクテーションキット向けバイナリの導入

キット内のバイナリはRaspberry Piでは動作しない

→先ほどビルドしたバイナリをbin/linuxにリンクすることで代替とする

```
$ rm bin/linux/*  
$ ln -s /usr/local/bin/adinrec bin/linux  
$ ln -s /usr/local/bin/adintool bin/linux  
$ ln -s /usr/local/bin/julius bin/linux  
$ ln -s /usr/local/bin/jcontrol bin/linux
```

※後述の run-linux-gmm.sh を、導入済みバイナリを使うように書き換えてもいい

```
#!/bin/sh  
julius -C main.jconf -C am-gmm-jconf -demo $*
```

■ デモコードの実行

・ GMMでのデモコード

```
$ cd /home/pi/julius/dictation-kit-4.5
$ export ALSADEV="plughw:1,0"
$ sh run-linux-gmm.sh
<中略>
<<< please speak >>>
```

「衆院議員は、具体的にどう考えているのか」

※最初の1文は認識率が悪い：仕様

・ DNNでのデモコード

```
$ export ALSADEV="plughw:1,0"
$ sh run-linux-dnn.sh
<中略>
<<< please speak >>>
```

「衆院議員は、具体的にどう考えているのか」

※Raspberry Piだと一応動作するが、起動も解析も、ものすごく遅い

■ディクテーションキットをつかった認識は重い

- ・デモコードでは、文章の解析を
「現代日本語書き言葉均衡コーパス」(約一億語)から生成した
をつかった言語モデル/発音辞書(語彙約6万語)を用いている

```
-d model/lang_m/bccwj.60k.bigram  
-v model/lang_m/bccwj.60k.htkdic
```

- ・自由文じゃなくて、認識させたい語彙のみを定義した辞書をつくることで、
認識速度や認識率を向上させることができる

■辞書の種類

- ・単語辞書(孤立単語認識)：単語辞書に当てはめて解析 (-w)
- ・記述文法：決められた文法による解析（「xxをn個ください」文法を定義）(-gram)
- ・単語N-gram：辞書を使わず形態素解析による単語切り出し (-d -v)

■ 認識させたい語句を書いた読みファイルから、ツールを使って辞書を作成

・ 読みファイル形式

文字コード：UTF-8 ※Julius 4.5から文字コードはUTF-8に統一

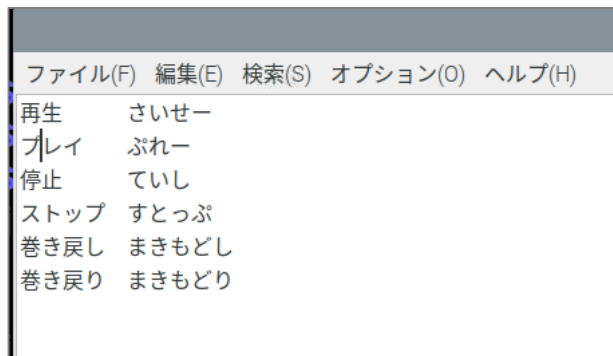
改行コード：LF

単語の区切り：タブ

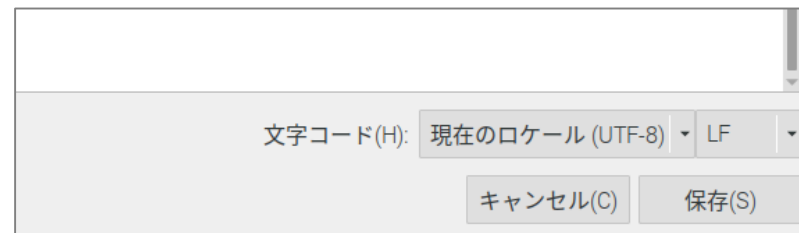
拡張子：.yomi

空行は入れない

RaspbianのText Editor(アクセサリ→Text Editor)の作成例



ファイル→別名で保存



※/home/pi/julius に保存しておく (sample.yomi)

■ 読みファイルから単語辞書の変換

- yomi2voca.pl

標準出力に辞書ができるので、ファイルに出力する

```
$ cd /home/pi/julius  
$ yomi2voca.pl sample.yomi > sample.dict
```

```
pi@raspberrypi:~/julius $ yomi2voca.pl sample.yomi > sample.dict  
pi@raspberrypi:~/julius $ cat sample.dict  
再生      s a i s e:  
プレイ    p u r e:  
停止      t e i s h i  
ストップ      s u t o q p u  
巻き戻し      m a k i m o d o s h i  
巻き戻り      m a k i m o d o r i  
pi@raspberrypi:~/julius $
```

■ 単語辞書を使ってみる

- ・ 音響モデルはディクテーションキットのもの
(/home/pi/Julius/dictation-kit-4.5/mode/phone_m/) を使う

/home/pi/julius/mydict.sh を作成

```
#!/bin/sh
/usr/local/bin/julius \
  -h /home/pi/julius/dictation-kit-4.5/model/phone_m/jnas-tri-3k16-gid.binhmm \
  -hlist /home/pi/julius/dictation-kit-4.5/model/phone_m/logicalTri \
  -input mic \
  -48 \
  -w $1
```

mydict.sh 実行例 :

```
$ cd /home/pi/julius
$ sh mydict.sh sample.dict
```

```
### read waveform input
Stat: adin_alsa: device name from ALSADEV: "plughw:1,0"
Stat: capture audio at 48000Hz
Stat: adin_alsa: latency set to 32 msec (chunk = 1536 bytes)
Stat: "plughw:1,0": Device [USB PnP Sound Device] device US
STAT: AD-in thread created
pass1_best: 巻き戻り
pass1_best_wordseq: 巻き戻り
pass1_best_phonemeseq: silB m a k i m o d o r i silE
pass1_best_score: -2062.172119
sentence1: 巻き戻り
wseq1: 巻き戻り
phseq1: silB m a k i m o d o r i silE
cmscore1: 0.676
score1: -2062.172119
<<< please speak >>>^C
```

■ 文法による提携認識

- ・ 長文の認識が効率的になり、引数をつかった処理が可能となる
- ・ ユースケース：対話システム、音声制御

■ 文法定義に必要なファイル：文法規則ファイル(grammar)、語彙辞書(yomi,voca)

- ・ ツールをつかって.dict/.dfa / .term ファイルに変換する

■ サンプル文法辞書の要件：機器制御

[機器] [制御]

[機器] 照明 / エアコン / LED

[制御] オン / オフ / 運転 / 停止

■ 文法規則ファイル (.grammer)

```
S          : NS_B KIKI_N CONTROL_N NS_E
KIKI_N     : KIKI
CONTROL_N  : CONTROL_ON
CONTROL_N  : CONTROL_OFF
```

文字コード : UTF-8
改行コード : LF
単語の区切り : タブ
拡張子 : .yomi
空行は入れない

※ /home/pi/juliusに
sample2.grammer
sample2.yomi
を保存しておく

■ 語彙ファイル(.yomi)

```
%KIKI
照明      しょーめー
照明      しょうめー
照明      しょーめい
照明      しょうめい
エアコン  えあこん
LED       えるいーでい
LED       えるいーでー
LED       えるいーでいー
%CONTROL_ON
つけて    つけて
つける    つける
オン      おん
運転      うんてん
%CONTROL_OFF
けして    けして
けす      けす
オフ      おふ
停止      ていし
%NS_B
<s>       silB
%NS_E
</s>      silE
```


■ 読みファイルから語彙ファイルへの変換

```
$ cd /home/pi/julius  
$ yomi2voca.pl sample2.yomi > sample2.voca
```

■ 文法/語彙ファイルから辞書(.dict)、オートマトン(.dfa)、対応ファイル(.term)を作る

```
$ mkdafa.pl sample2  
$ ls sample2.*  
sample2.dfa sample2.dict sample2.grammar sample2.term sample2.voca sample2.yomi
```

■ 文法ファイルを使って認証してみる

- /home/pi/julius/mydict2.sh を

```
pi@raspberrypi:~/julius $ cat mydict2.sh
#!/bin/sh
/usr/local/bin/julius \
  -h /home/pi/julius/dictation-kit-4.5/model/phone_m/jnas-tri-3k16-gid.binhmm \
  -hlist /home/pi/julius/dictation-kit-4.5/model/phone_m/logicalTri \
  -input mic \
  -48 \
  -gram /home/pi/julius/sample2
```

拡張子を指定しない：同じディレクトリの.dict .dfa .term を使う

```
----- System Information end -----
Notice for feature extraction (01),
*****
* Cepstral mean normalization for real-time decoding:      *
* NOTICE: The first input may not be recognized, since     *
*   no initial mean is available on startup.                *
*****
-----
I
### read waveform input
Stat: adin_alsa: device name from ALSADEV: "plughw:1,0"
Stat: capture audio at 48000Hz
Stat: adin_alsa: latency set to 32 msec (chunk = 1536 bytes)
Stat: "plughw:1,0": Device [USB PnP Sound Device] device USB Audio [USB Audio] subdevice #0
STAT: AD-in thread created
<<< please speak >>>
```

- 辞書にむりやり割り当てようとする：低い認識率でもなにか結果を返す
認識結果を使う（次項）場合は、出力される評価スコアによる判断が必要。

```
pass1_best: ストップ  
pass1_best_wordseq: ストップ  
pass1_best_phonemeseq: silB s u t o q p u silE  
pass1_best_score: -2410.181885  
sentence1: ストップ  
wseq1: ストップ  
phseq1: silB s u t o q p u silE  
cmscore1: 0.892  
score1: -2410.181885
```

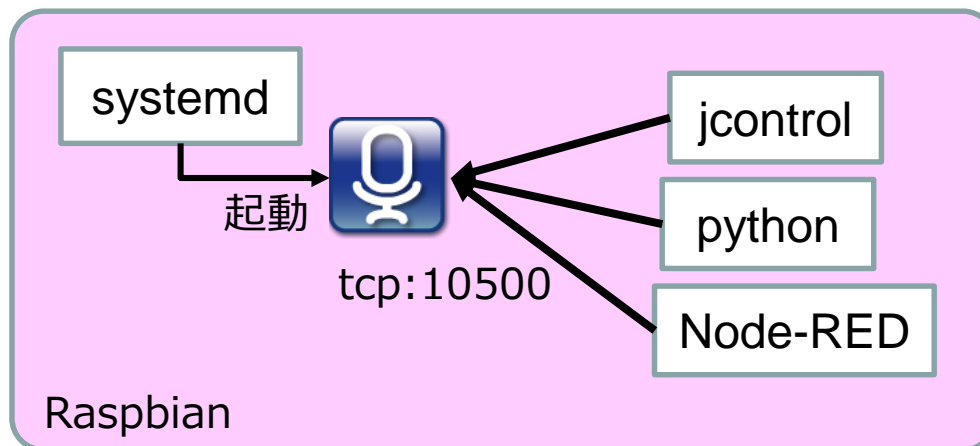
```
pass1_best: プレイ  
pass1_best_wordseq: プレイ  
pass1_best_phonemeseq: silB p u r e: silE  
pass1_best_score: -3751.027832  
sentence1: プレイ  
wseq1: プレイ  
phseq1: silB p u r e: silE  
cmscore1: 0.939  
score1: -3751.027832
```

■ 認識結果を活用するための工夫

- ・ juliusコマンドは認識結果を標準出力する
→ シェルスクリプトからjuliusコマンドを呼び出して、標準出力をgrepして認識結果を使うことが可能

- ・ juliusをサービスとして起動する

Raspbianの標準サービスフレームワーク「Systemd」を使い、Juliusの音声認識をサービス（モジュールモード）として常駐
→ 標準添付されているjcontrolコマンドを使ったり、
シェルスクリプトやPython、Node-REDから音声認識を使える



■ モジュールモードの設定ファイルの作成

- /home/pi/julius/mydict-mod.jconf の作成

```
-h /home/pi/julius/dictation-kit-4.5/model/phone_m/jnas-tri-3k16-gid.binhmm
-hlist /home/pi/julius/dictation-kit-4.5/model/phone_m/logicalTri
-input mic
-48
-w /home/pi/julius/sample.dict
-module
```

※辞書は単語辞書(P.20)を使用

- 試しにモジュールモードで実行

```
$ julius -C /home/pi/julius/mydict-mod.jconf
```

```
STAT: [3] initialize for acoustic HMM calculation
Stat: outprob_init: state-level mixture PDFs, use calc_mix()
Stat: addlog: generating addlog table (size = 1953 kB)
Stat: addlog: addlog table generated
STAT: [4] prepare MFCC storage(s)
STAT: [5] prepare for real-time decoding
STAT: All init successfully done

Stat: server-client: socket ready as server
////////////////////////////////////
///  Module mode ready
///  waiting client at 10500
////////////////////////////////////
```

Ctrl-Cで停止

■ サービス実行定義ファイルを作成

・ julius.serviceの作成

```
[Unit]
Description=Julius

[Service]
Type=simple
ExecStart=/usr/local/bin/julius -C /home/pi/julius/mydict-mod.jconf
Restart=always
Environment=ALSADEV="plughw:1,0"

[Install]
WantedBy=default.target
```

・ julius.serviceの配置

```
$ sudo cp julius.service /etc/systemd/system/
```

※ 辞書変更等で /etc/systemd/system/julius.service を変更した場合は、
sudo systemctl daemon-reload
を実行する必要がある。

■ サービス実行と確認

- julius.serviceの実行

```
$ sudo systemctl start julius.service
```

※ /etc/systemd/system/julius.service を変更した場合は、
sudo systemctl daemon-reload
を実行する必要がある。

- julius.serviceの確認

```
$ systemctl status julius.service
```

正常

```
pi@raspberrypi:~/julius $ sudo systemctl status julius.service
• julius.service - Julius
  Loaded: loaded (/etc/systemd/system/julius.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2019-08-22 12:35:25 JST; 3min 0s ago
    Main PID: 15475 (julius)
      Tasks: 2 (limit: 2200)
     Memory: 18.6M
        CGroup: /system.slice/julius.service
                └─15475 /usr/local/bin/julius -C /home/pi/julius/mydict-mod.jconf
```

異常

```
pi@raspberrypi:~/julius $ sudo systemctl status julius.service
• julius.service - Julius
  Loaded: loaded (/etc/systemd/system/julius.service; enabled; vendor preset: e
  Active: failed (Result: exit-code) since Thu 2019-08-22 12:33:18 JST; 14s ago
         http://git.io/wiopi
```

■ julius.serviceの停止

```
$ sudo systemctl stop julius.service
```

■ 起動時に自動実行

```
$ sudo systemctl enable julius.service
```

■ 起動時の自動実行をやめる

```
$ sudo systemctl disable julius.service
```

■ モジュールモードでの注意：

- ・ 同時に複数クライアントからの接続はできません

- julius標準のクライアント「jcontrol」でサービスの動作を確認

```
$ jcontrol localhost 10500
```

```
pi@raspberrypi:~ $ jcontrol localhost 10500
connecting to localhost:10500...done
> <STARTPROC/>
> <INPUT STATUS="LISTEN" TIME="1566446757"/>
> <INPUT STATUS="STARTREC" TIME="1566446811"/>
> <STARTRECOG/>
> <INPUT STATUS="ENDREC" TIME="1566446812"/>
> <ENDRECOG/>
> <INPUTPARAM FRAMES="63" MSEC="630"/>
> <RECOGOUT>
>   <SHYPO RANK="1" SCORE="-1631.896729" GRAM="0">
>     <WHYPO WORD="再生" CLASSID="再生" PHONE="silB s a i s e: silE" CM="0.999"/>
>   </SHYPO>
> </RECOGOUT>
> <INPUT STATUS="LISTEN" TIME="1566446812"/>
```

各行先頭に > の入ったXML形式でレスポンスが得られる

■ 単語辞書(P.20)を想定したスクリプト : sample.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import socket
import xml.etree.ElementTree as ET

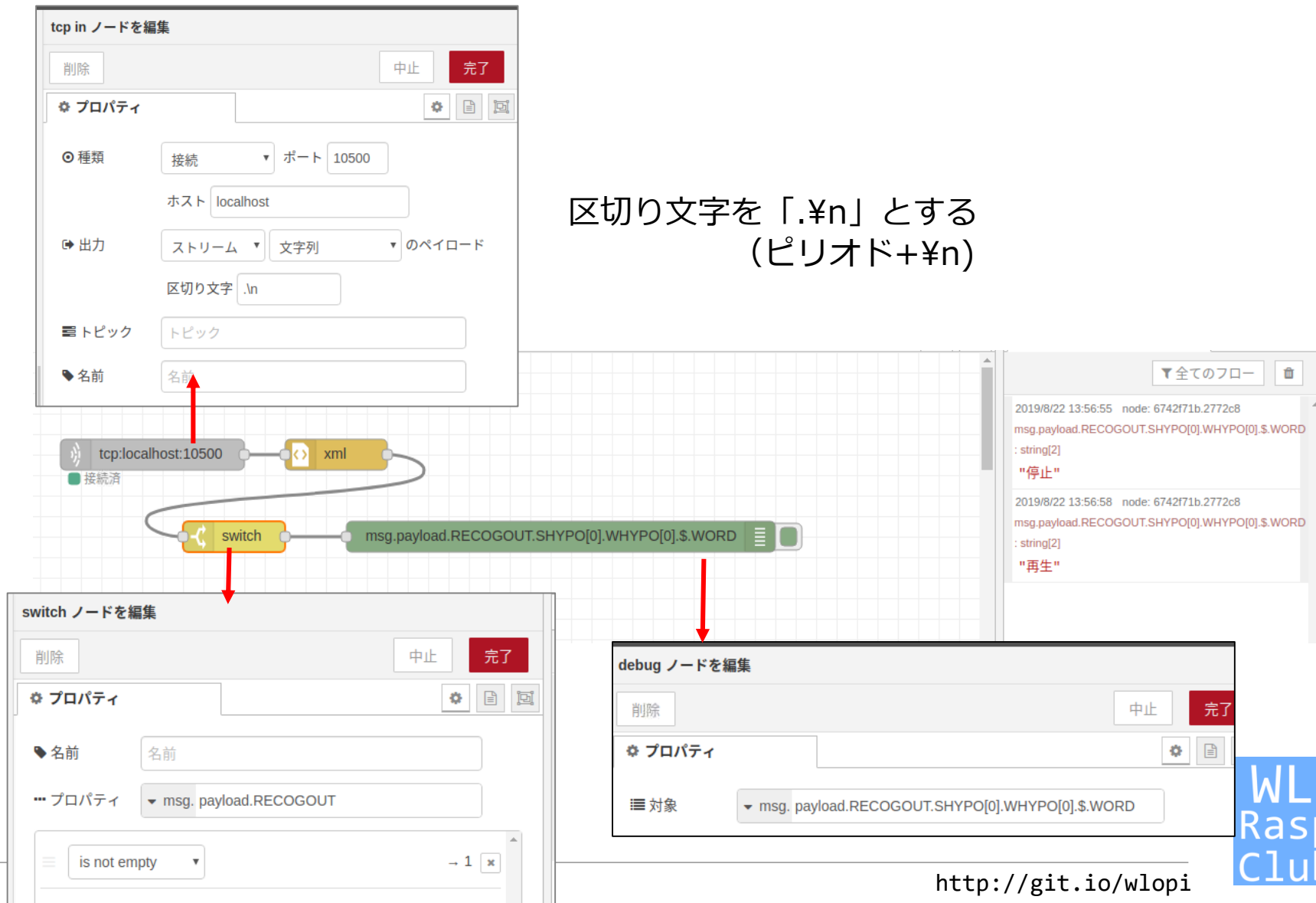
def main():
    host = 'localhost'
    port = 10500
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    try:
        data = ''
        while 1:
            if '</RECOGOUT>¥n.' in data:
                root = ET.fromstring('<?xml version="1.0"?>¥n' + data[data.find('<RECOGOUT>'):].replace('¥n.', ''))
                for wp in root.findall('./SHYPO/WHYPO'):
                    command = wp.get('WORD')
                    score = float(wp.get('CM'))
                    if command == u'再生' and score >= 0.9:
                        print u'play'
                    elif command == u'プレイ' and score >= 0.996:
                        print u'play'
                    elif command == u'停止' and score >= 0.996:
                        print u'stop'
                    elif command == u'ストップ' and score >= 0.996:
                        print u'stop'
                    elif command == u'巻き戻し' and score >= 0.996:
                        print u'rew'
                    elif command == u'巻き戻り' and score >= 0.996:
                        print u'rew'
                data = ''
            else:
                data = data + client.recv(1024)
    except KeyboardInterrupt:
        client.close()

if __name__ == "__main__":
    main()
```

実行は
\$ python sample.py

■ 単語辞書(P.20)を想定したFlow : julius-nodered-sample.json



■ いいマイクをつかう

- ・ 指向性を持つ、コンデンサーマイクを使う
- ・ 会議用のマイクを使う
- ・ マイクと口との距離

■ いい声で

- ・ 大きな声ではきはきと
- ・ 高すぎても低すぎても、早すぎても遅すぎてもだめ

■ ALSAの設定

- ・ マイクの設定は大きすぎても音が割れてしまう (80-90)
- ・ adintoolの録音品質を確認

■ juliusの設定

- ・ 検出しきい値 (-lv)
- ・ 音響尤度(ゆうど)計算値 (-tmix)

■ Juliusサイト

- ・ 公式サイト <https://julius.osdn.jp/>
- ・ 開発リポジトリ <https://github.com/julius-speech/julius>
- ・ juliusbook <https://julius.osdn.jp/index.php?q=documents.html#juliusbook>

■ 論文・解説

Juliusを用いた音声認識インタフェースの作成

李 晃伸（名古屋工業大学） 河原達也（京都大学）

<http://sap.ist.i.kyoto-u.ac.jp/members/kawahara/paper/RI-HIS09.pdf>

■ ブログ等

systemd - 自作プログラムをサービス化して自動起動しよう- こんぺき草紙

<https://takayu90tech.com/category/julius>

音声認識してみよう - ベゼリーのページ

<http://bezelie.com/flitz/音声合成してみる/>

Qiita - juliusタグのついた記事

<https://qiita.com/tags/julius>