MacspeechX.py MODULE AND ITS USE IN AN ACCELERATOR CONTROL SYSTEM

Noboru Yamamoto*, J-PARC cener, KEK and JAEA, Ibaraki, JAPAN

Abstract

macspeechX.py[1] is a Python module to accels speech synthesis library on MacOSX. This module have been used in the vocal alert system in KEKB[2] and J-PARC[3] accelerator control system. Recent upgrade of this module allow us to handle non-English lanugage, such as Japanese, through this module. Implementation detail will be presented as an example of Python program accessing system library.

SPEECH SYNTHESIS IN CONTROL SYSTEMS

In some control system, alerts to the operators can be sent as vocal messages. It used be require the special hardware or software to generate vocal message from computers in the system.

When we started commissioning of KEKB accelerator, such an alert system was requested. We picked up:

- speech synthesis library includes as one of standard libraries on Macintosh OS from Apple.
- Macspeech.py module distributed as one of standard module with Python programming Langauge

With these two components, we could build a very low cost but flexible vocal alert system. This system has been used in KEKB control system and J-PARC control system. During the operation of the accelerators, we needed as little modification in the maing program. On the other hand changes in the software & hardware environment, forced us to develop a new python module, macspeechX.py. This module evolves to current status following the evolution of software/hardware.

We will describes the evolution of this modules in this article.

KEKB/J-PARC VOCAL ALERT SYSTEM

The main components of the KEKB/J-PARC vocal alert system is a macintosh computer and the tiny python program running on this macintosh. A main task of this tiny program is simply

1. Wait for UDP packet including plain text from the control system at multiple upb ports,
2. Pass this message to speech synthesis library.

With additional functionality such as user interface or selection of voices for specified UDP ports, this program can fit one or two pages of the paper

While this system running without serious problem until MacOSX came to the market. In Python on MacOSX does not includes macspeech.py as a its components. It means we need to develop our own solution before old Mac hardware would be replaced by new hardware which just runs MacOSX.

In the next section, we will see several ways to write Python module which bridges C/C++ library.

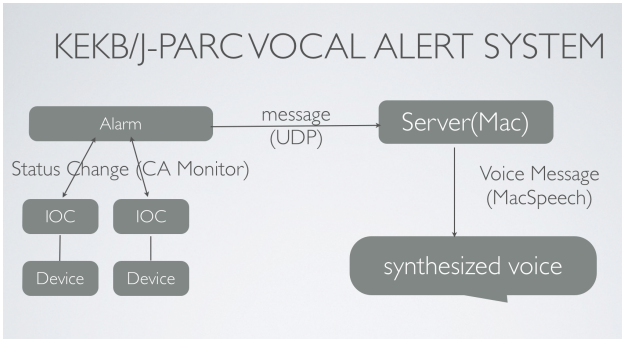

Figure 1: Software overview of KEKB/J-PARC vocal alert system.

HOW TO WRITE PYTHON EXTENSION MODULE

There are several ways to prepare python extension modules which allow access to the existing C-library from Python.

1. Write a glue module in C and/or C++ using Python API.
2. Use SWIG to generate a glue code automatically from a configuration file.
3. Use cython to generate glue module from cython description of the library.
4. Use ctypes module to develop a glue module in Python language
5. Use PyObjc, which is available only on MacOSX and Darwin operating system, for glue module writen in Python.

*noboru.yamamoto@kek.jp

ISBN 978-3-95450-139-7

Python C API

Python, also know as CPython[4], defines a set of APIs to access internal structures of Python interpreter. Python C modules can be developed using these APIs. KEK version of EPICS CA[9] module uses this method. It is most powerfull and efficient way to bridge C/C++ library with Python. Careful coding is required especially in multi threaded environment, otherwise you can crash your system, quite easily.

SWIG(Simple Wrapper Interface Generator)

SWIG[5] is a tool developed by Dave Beazley. It reads C/C++ header file like description of library and output files to glue this library to a target language. It support many script-type languages such as Tcl/Tk, Java, Perl, Ruby and Python and much more. If you would like to have integrate your or someone else's library with many script languages, SWIG can realise it with minimum effort.

You can start from C/C++ header files as a SWIG input file and will need to modify it for better integration. You must pay attension to header files and SWIG configuration files. You also need to follow upgrade of SWIG by yourself.

Cython

In cython, you writes Python-like description of C/C++ library. Cython[6] convert it to C/C++ glue codes and python support modules. Compiled C/C++ glue codes can be imported from Python interpreter. Cython (not Cpython) define a language similar to Python but with type declaration for efficient code generation.

ctypes

"ctypes"[7] is now one of Python standard module. It allows access to dynamic loadable library from Python interpreter without compilation and linking of the code. You need to provide a python module to access your target library in Python/ctypes way.

Ctypes is powerful. It may be too powerful, so that you can crash your system from Python program. You also need to convert information in C/C++ header files, such as name and type of class members, to Python/ctypes description by yourself.

PyObjc

PyObjc[8] is only available MacOSX/Darin environment. It provieds way to access Cocoa library from Python program. Mapping of the name in Python and it in the Cocoa library is created dyanmically, so you don't need to provide mapping rules by yourself.

It is easiest way to access Cocoa or Objective-C libraries in the MacOSX/Darwin environment. Apparently, you cannot port it to other platforms.

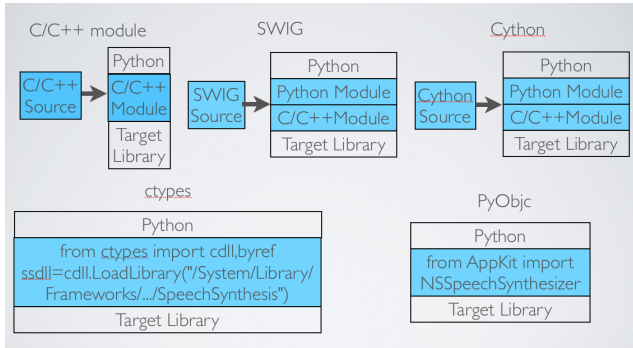

Figure 2: Various Methods to extend Python using modules linking to external libraries.

MACSPEECHX.PY MODULE

macspeech.py

The macspeech.py is a python module to access speech synthesis library on Mac OS written by the author of Python programming language, Guild von Rossum, himself. It has simple API, such as SpeakString() function, to access library and provides a few classes for advanced use of the library. The vocal alert system described above uses these classes.

This module was one of standard modules in Python distribution for Macintosh OS. It gradually faded away from the distribution and provided only for MacOS upto version 9.

Macspeech.py to macspeechX.py

After the introduction of Mac OSX operating system, the macspeech.py module was dropped from Python distribution which supports MacOSX.

On the other hand, MacOSX itself include improved speech synthesis library. So if we have a python extension module which has same API with the original macspeech.py, we can continue to use our vocal alert system with minimum modification, changing the name of imported module in the python program.

The first version of macspeechX.py was written in 2005 using the ctypes module to access MacOSX speech synthesis library through Carbon API(or C/C++API). It provides APIs compatible with the macspeech.py module so that user of macpsech.py can simply replace macspeech with macspeechX. In the extension module using ctypes, it is necessary to define classes corresponding C-classes in the target library, even if you don't need to access them from Python side to complete interface definitions of functions/methods in the library. We choose the cytypes approach among several methods described in the previous section, because of ease of development. We already had some experience with ctypes at that time and PyObjC was not a standard part of the system yet.

This version of macspeechX also support use of embedded speech commands in a message passed to MacOSX speech synthesis library. Embedded speech

ISBN 978-3-95450-139-7

commands include phonetic symbols to represents spoken message. We can imitate Japanese using these phonetic symbols, in principle.

Current Version

Since MacOSX 10.7, Carbon API for speech synthesis library became obsolete and only Cocoa(Obj-C) APIs are provided. This is a version which officially supports multiple language, including Japanese, in speech synthesis library. The latest version of macspeechX.py now uses both ctypes and PyObjc to access underling system library. It also provide several new API to access new functionality in the library. Most importantly for us is the support of Japanese language. This module should works with any languages supported by MacOSX and it speech synthesis library.

Future Release

When the first version of macspeechX.py was developed, ctypes was one of standard libraries in Python bet PyObjC. It become a part of MacOSX when Apple released MacOSX 10.5 Leopard in 2007. It means every Macs shipped from Apple includes PyObjC and Python and ready to work. In the future releases of macspeechX.py, only PyObjc will be used to access underling speech synthesis library. It will reduce a size of code and will have higher maintainability. Test version of macspeechX.py in PyObjc has a half size of the current version of macspeechX.py

REFERENCES

[1] macspeechX.py is available on the web at https://pypi.python.org/pypi/macspeechX/1.1a .

[2] S-I Kurokawa et al. “Control System Design for KEKB Accelerators”, in the proceedings of PAC'95, p.2205; http://accelconf.web.cern.ch/AccelConf/p95/ARTICLES/MPA/MPA06.PDF

[3] J-PARC Web site ; http://www.j-parc.jp/

[4] “Python/C API Reference Manual”; http://docs.python.org/2/c-api/

[5] “SWIG” home page; http://www.swig.org

[6] “Cython: C-Extensions for Python”; http://docs.python.org/2/c-api/

[7] “ctypes – A foreign function library for Python”; http://docs.python.org/2/library/ctypes.html

[8] “PyObjC- the Python Objective-C bridge”; http://pythonhosted.org/pyobjc/

[9] “History” section of “New Python EPICS Interface” by Xiao-Qiang Wang; http://controls.web.psi.ch/cgi-bin/twiki/view/Main/NewPythonEpicsInterface#History

ISBN 978-3-95450-139-7