JCC Installation Instructions

Table of contents

1. Getting JCC's Source Code

JCC's source code is included with PyLucene's. If you've downloaded the PyLucene source code already, JCC's is to be found in the `jcc` subdirectory.

To get the JCC source code only from SVN use:

```
$ svn co
http://svn.apache.org/repos/asf/lucene/pylucene/trunk/jcc jcc
```

2. Building JCC

JCC is a Python extension written in Python and C++. It requires a Java Runtime Environment to operate as it uses Java's reflection APIs to do its work. It is built and installed via `distutils` or setuptools.

1. Edit `setup.py` and review that values in the `INCLUDES`, `CFLAGS`, `DEBUG_CFLAGS`, `LFLAGS` and `JAVAC` are correct for your system. These values are also going to be compiled into JCC's `config.py` file and are going to be used by JCC when invoking `distutils` or `setuptools` to compile extensions it is generating code for.
2. At the command line, enter:

```
$ python setup.py build
$ sudo python setup.py install
```

3. Requirements

JCC requires a Java Development Kit to be present. It uses the Java Native Invocation Interface and expects `<jni.h>` and the Java libraries to be present at build and runtime.

JCC requires a C++ compiler. A recent C++ compiler for your platform is expected to work as expected.

4. Shared Mode: Support for the --shared Flag

JCC includes a small runtime that keeps track of the Java VM and of Java objects escaping it. Because there can be only one Java VM embedded in a given process at a time, the JCC runtime must be compiled as a shared library when more than one JCC-built Python extension is going to be imported into a given Python process.

Shared mode depends on `setuptools`' capability of building plain shared libraries (as opposed to shared libraries for Python extensions). This shared library capability is a feature currently under development.

Currently, shared mode is supported with `setuptools 0.6c7` and above out of the box on Mac OS X and Windows. On Linux, a patch to `setuptools` needs to be applied first. This patch is included in the JCC source distribution in the `jcc/patches` directory, `patch.43`. This patch was submitted to the `setuptools` project via issue 43.

The `shared mode disabled` error reported during the build of JCC's on Linux contains the exact instructions on how to patch the `setuptools` installation with `patch.43` on your system.

Shared mode is also required when embedding Python in a Java VM as JCC's runtime shared library is used by the JVM to load JCC and bootstrap the Python VM via the JNI.

When shared mode is not enabled, not supported or `distutils` is used instead of `setuptools`, static mode is used instead. The JCC runtime code is statically linked with eacg JCC-built Python extension and only one such extension can be used in a given Python process.

As setuptools grows its shared library building capability it is expected that more operating systems should be supported with shared mode in the future.

Shared mode can be forced off by building JCC with the `NO_SHARED` environment variable set.

There are two defaults to consider here:

- Is JCC built with shared mode support or not ?
  - By default, on Mac OS X and Windows this is the case
  - By default, on Linux, this is the case. if `setuptools` is patched.
  - On other operating systems shared mode support is off by default - not supported - because shared mode depends on `setuptools`'s capability of building a regular shared library which is still an experimental feature.
- Is a JCC-built Python extension built with shared mode ?
  By default, no, shared mode is enabled only with the `--shared` command line argument.

5. Notes for Mac OS X

On Mac OS X, Java is installed by Apple's setup as a framework. The values in `setup.py` for `INCLUDES` and `LFLAGS` for `darwin` should be correct and ready to use.

6. Notes for Linux

JCC has been built and tested on a variety of Linux distributions, 32- and 64-bit. Getting the

java configuration correct is important and is done differently for every distribution.
For example:

- on Ubuntu, to install Java 5, these commands may be used:

```
$ sudo apt-get install sun-java5-jdk
$ sudo update-java-alternatives -s java-1.5.0-sun
```

  The samples flags for Linux in JCC's setup.py should be close to correct.
- on Gentoo, the `java-config` utility should be used to locate, and possibly change, the default java installation. The sample flags for Linux in JCC's `setup.py` should be changed to reflect the root of the Java installation which may be obtained via:

```
$ java-config -O
```

See earlier section about Shared Mode for Linux support.

7. Notes for Solaris

At this time, JCC has been built and tested only on Solaris 11 with Sun Studio C++ 12, Java 1.6 and Python 2.4.

Because JCC is written in C++, Python's `distutils` must be nudged a bit to invoke the correct compiler. Sun Studio's C compiler is called `cc` while its C++ compiler is called `CC`. To build JCC, use the following shell command to ensure that the C++ compiler is used:

```
$ CC=CC python setup.py build
```

Shared mode is not currently implemented for Solaris, `setuptools` needs to be taught how to build plain shared libraries on Solaris first.

8. Notes for Windows

At this time, JCC has been built and tested on Win2k and WinXP with a variety of Python and Java versions.

- Adding the Python directory to `PATH` is recommended.
- Adding the Java directories containing the necessary DLLs and to `PATH` is a must.
- Adding the directory containing `javac.exe` to `PATH` is required for shared mode (enabled by default if `setuptools >= 0.6c7` is found to be installed).

9. Notes for Python 2.3

To use JCC with Python 2.3, setuptools is required

1. download setuptools.
2. edit the downloaded `setuptools` egg file to use python2.3 instead of python2.4.
3. At the command line, run:
   ```
   $ sudo sh setuptools-0.6c7-py2.4.egg
   ```