Building C and C++ Extensions

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 1)

Unknown directive type "highlight".

```
.. highlight:: c
```

A C extension for CPython is a shared library (e.g. a `.so` file on Linux, `.pyd` on Windows), which exports an *initialization function*.

To be importable, the shared library must be available on :envvar:`PYTHONPATH`, and must be named after the module name, with an appropriate extension. When using distutils, the correct filename is generated automatically.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 12); *backlink*

Unknown interpreted text role "envvar".

The initialization function has the signature:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 18)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyInit_modulename(void)
```

It returns either a fully-initialized module, or a :c:type:`PyModuleDef` instance. See :ref:`initializing-modules` for details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 20); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 20); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 23)

Unknown directive type "highlight".

```
.. highlight:: python
```

For modules with ASCII-only names, the function must be named `PyInit_<modulename>`, with `<modulename>` replaced by the name of the module. When using :ref:`multi-phase-initialization`, non-ASCII module names are allowed. In this case, the initialization function name is `PyInitU_<modulename>`, with `<modulename>` encoded using Python's *punycode* encoding with hyphens replaced by underscores. In Python:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 25); *backlink*

Unknown interpreted text role "ref".

```
def initfunc_name(name):
    try:
        suffix = b'_' + name.encode('ascii')
    except UnicodeEncodeError:
        suffix = b'U_' + name.encode('punycode').replace(b'-', b'_')
    return b'PyInit' + suffix
```

It is possible to export multiple modules from a single shared library by defining multiple initialization functions. However, importing them requires using symbolic links or a custom importer, because by default only the function corresponding to the filename is found. See the *"Multiple modules in one library"* section in PEP 489 for details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 46)

Unknown directive type "highlight".

```
.. highlight:: c
```

Building C and C++ Extensions with distutils

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 51)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Martin v. LÃ¶wis <martin@v.loewis.de>
```

Extension modules can be built using distutils, which is included in Python. Since distutils also supports creation of binary packages, users don't necessarily need a compiler and distutils to install the extension.

A distutils package contains a driver script, :file:`setup.py`. This is a plain Python file, which, in the most simple case, could look like this:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 57); *backlink*

Unknown interpreted text role "file".

```
from distutils.core import setup, Extension

module1 = Extension('demo',
                    sources = ['demo.c'])

setup (name = 'PackageName',
       version = '1.0',
       description = 'This is a demo package',
       ext_modules = [module1])
```

With this :file:`setup.py`, and a file :file:`demo.c`, running

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 73); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 73); *backlink*

Unknown interpreted text role "file".

```
python setup.py build
```

will compile :file:`demo.c`, and produce an extension module named `demo` in the :file:`build` directory. Depending on the system, the module file will end up in a subdirectory :file:`build/lib.system`, and may have a name like :file:`demo.so` or :file:`demo.pyd`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 77); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 77); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 77); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 77); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 77); *backlink*

Unknown interpreted text role "file".

In the :file:`setup.py`, all execution is performed by calling the `setup` function. This takes a variable number of keyword arguments, of which the example above uses only a subset. Specifically, the example specifies meta-information to build packages, and it specifies the contents of the package. Normally, a package will contain additional modules, like Python source modules, documentation, subpackages, etc. Please refer to the distutils documentation in :ref:`distutils-index` to learn more about the features of distutils; this section explains building extension modules only.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 82); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 82); *backlink*

Unknown interpreted text role "ref".

It is common to pre-compute arguments to :func:`setup`, to better structure the driver script. In the example above, the `ext_modules` argument to :func:`~distutils.core.setup` is a list of extension modules, each of which is an instance of the :class:`~distutils.extension.Extension`. In the example, the instance defines an extension named `demo` which is build by compiling a single source file, :file:`demo.c`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 91); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 91); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 91); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 91); *backlink*

Unknown interpreted text role "file".

In many cases, building an extension is more complex, since additional preprocessor defines and libraries may be needed. This is demonstrated in the example below.

```
from distutils.core import setup, Extension

module1 = Extension('demo',
                    define_macros = [('MAJOR_VERSION', '1'),
                                     ('MINOR_VERSION', '0')],
                    include_dirs = ['/usr/local/include'],
                    libraries = ['tcl83'],
                    library_dirs = ['/usr/local/lib'],
                    sources = ['demo.c'])

setup (name = 'PackageName',
       version = '1.0',
       description = 'This is a demo package',
       author = 'Martin v. Loewis',
       author_email = 'martin@v.loewis.de',
       url = 'https://docs.python.org/extending/building',
       long_description = '''
This is really just a demo package.
''',
       ext_modules = [module1])
```

In this example, :func:`~distutils.core.setup` is called with additional meta-information, which is recommended when distribution packages have to be built. For the extension itself, it specifies preprocessor defines, include directories, library directories, and libraries. Depending on the compiler, distutils passes this information in different ways to the compiler. For example, on Unix, this may result in the compilation commands

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 127); *backlink*

Unknown interpreted text role "func".

```
gcc -DNDEBUG -g -O3 -Wall -Wstrict-prototypes -fPIC -DMAJOR_VERSION=1 -DMINOR_VERSION=0 -I/usr/local/include -I/usr/local/include/python2
```

```
gcc -shared build/temp.linux-i686-2.2/demo.o -L/usr/local/lib -ltcl83 -o build/lib.linux-i686-2.2/demo.so
```

These lines are for demonstration purposes only; distutils users should trust that distutils gets the invocations right.

Distributing your extension modules

When an extension has been successfully built, there are three ways to use it.

End-users will typically want to install the module, they do so by running

```
python setup.py install
```

Module maintainers should produce source packages; to do so, they run

```
python setup.py sdist
```

In some cases, additional files need to be included in a source distribution; this is done through a :file:`MANIFEST.in` file; see :ref:`manifest` for details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 158); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)building.rst`, line 158); *backlink*

Unknown interpreted text role "ref".

If the source distribution has been built successfully, maintainers can also create binary distributions. Depending on the platform, one of the following commands can be used to do so.

```
python setup.py bdist_rpm
python setup.py bdist_dumb
```