Building C and C++ Extensions on Windows

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

Unknown directive type "highlight".

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

This chapter briefly explains how to create a Windows extension module for Python using Microsoft Visual C++, and follows with more detailed background information on how it works. The explanatory material is useful for both the Windows programmer learning to build Python extensions and the Unix programmer interested in producing software which can be successfully built on both Unix and Windows.

Module authors are encouraged to use the distutils approach for building extension modules, instead of the one described in this section. You will still need the C compiler that was used to build Python; typically Microsoft Visual C++.

Note

This chapter mentions a number of filenames that include an encoded Python version number. These filenames are represented with the version number shown as `XY`; in practice, `'X'` will be the major version number and `'Y'` will be the minor version number of the Python release you're working with. For example, if you are using Python 2.2.1, `XY` will actually be `22`.

A Cookbook Approach

There are two approaches to building extension modules on Windows, just as there are on Unix: use the :mod:`distutils` package to control the build process, or do things manually. The distutils approach works well for most extensions; documentation on using :mod:`distutils` to build and package extension modules is available in :ref:`distutils-index`. If you find you really need to do things manually, it may be instructive to study the project file for the :source:`winsound <PCbuild/winsound.vcxproj>` standard library module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 36); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 36); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 36); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 36); *backlink*

Unknown interpreted text role "source".

Differences Between Unix and Windows

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 50)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Chris Phoenix <cphoenix@best.com>
```

Unix and Windows use completely different paradigms for run-time loading of code. Before you try to build a module that can be dynamically loaded, be aware of how your system works.

In Unix, a shared object (:file:`.so`) file contains code to be used by the program, and also the names of functions and data that it expects to find in the program. When the file is joined to the program, all references to those functions and data in the file's code are changed to point to the actual locations in the program where the functions and data are placed in memory. This is basically a link operation.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)windows.rst`, line 57); *backlink*
>
> Unknown interpreted text role "file".

In Windows, a dynamic-link library (:file:`.dll`) file has no dangling references. Instead, an access to functions or data goes through a lookup table. So the DLL code does not have to be fixed up at runtime to refer to the program's memory; instead, the code already uses the DLL's lookup table, and the lookup table is modified at runtime to point to the functions and data.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)windows.rst`, line 64); *backlink*
>
> Unknown interpreted text role "file".

In Unix, there is only one type of library file (:file:`.a`) which contains code from several object files (:file:`.o`). During the link step to create a shared object file (:file:`.so`), the linker may find that it doesn't know where an identifier is defined. The linker will look for it in the object files in the libraries; if it finds it, it will include all the code from that object file.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)windows.rst`, line 70); *backlink*
>
> Unknown interpreted text role "file".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)windows.rst`, line 70); *backlink*
>
> Unknown interpreted text role "file".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)windows.rst`, line 70); *backlink*
>
> Unknown interpreted text role "file".

In Windows, there are two types of library, a static library and an import library (both called :file:`.lib`). A static library is like a Unix :file:`.a` file; it contains code to be included as necessary. An import library is basically used only to reassure the linker that a certain identifier is legal, and will be present in the program when the DLL is loaded. So the linker uses the information from the import library to build the lookup table for using identifiers that are not included in the DLL. When an application or a DLL is linked, an import library may be generated, which will need to be used for all future DLLs that depend on the symbols in the application or DLL.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)windows.rst`, line 76); *backlink*
>
> Unknown interpreted text role "file".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)windows.rst`, line 76); *backlink*
>
> Unknown interpreted text role "file".

Suppose you are building two dynamic-load modules, B and C, which should share another block of code A. On Unix, you would *not* pass :file:`A.a` to the linker for :file:`B.so` and :file:`C.so`; that would cause it to be included twice, so that B and C would each have their own copy. In Windows, building :file:`A.dll` will also build :file:`A.lib`. You *do* pass :file:`A.lib` to the linker for B and C. :file:`A.lib` does not contain code; it just contains information which will be used at runtime to access A's code.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main) (Doc) (extending)windows.rst`, line 86); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 86); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 86); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 86); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 86); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 86); *backlink*

Unknown interpreted text role "file".

In Windows, using an import library is sort of like using `import spam`; it gives you access to spam's names, but does not create a separate copy. On Unix, linking with a library is more like `from spam import *`; it does create a separate copy.

Using DLLs in Practice

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 105)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Chris Phoenix <cphoenix@best.com>
```

Windows Python is built in Microsoft Visual C++; using other compilers may or may not work (though Borland seems to). The rest of this section is MSVC++ specific.

When creating DLLs in Windows, you must pass :file:`pythonXY.lib` to the linker. To build two DLLs, spam and ni (which uses C functions found in spam), you could use these commands:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 112); *backlink*

Unknown interpreted text role "file".

```
cl /LD /I/python/include spam.c ../libs/pythonXY.lib
cl /LD /I/python/include ni.c spam.lib ../libs/pythonXY.lib
```

The first command created three files: :file:`spam.obj`, :file:`spam.dll` and :file:`spam.lib`. :file:`Spam.dll` does not contain any Python functions (such as :c:func:`PyArg_ParseTuple`), but it does know how to find the Python code thanks to :file:`pythonXY.lib`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 119); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 119); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 119); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 119); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 119); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 119); *backlink*

Unknown interpreted text role "file".

The second command created :file:`ni.dll` (and :file:`.obj` and :file:`.lib`), which knows how to find the necessary functions from spam, and also from the Python executable.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 124); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 124); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 124); *backlink*

Unknown interpreted text role "file".

Not every identifier is exported to the lookup table. If you want any other modules (including Python) to be able to see your identifiers, you have to say `_declspec(dllexport)`, as in `void _declspec(dllexport) initspam(void)` or `PyObject _declspec(dllexport) *NiGetSpamData(void)`.

Developer Studio will throw in a lot of import libraries that you do not really need, adding about 100K to your executable. To get rid of them, use the Project Settings dialog, Link tab, to specify *ignore default libraries*. Add the correct :file:`msvcrtxx.lib` to the list of libraries.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\(cpython-main)(Doc)(extending)windows.rst, line 133); *backlink*

Unknown interpreted text role "file".