15.8 C和Python中的线程混用¶

问题¶

你有一个程序需要混合使用C、Python和线程， 有些线程是在C中创建的，超出了Python解释器的控制范围。 并且一些线程还使用了Python C API中的函数。

解决方案¶

如果你想将C、Python和线程混合在一起，你需要确保正确的初始化和管理Python的全局解释器锁（GIL）。 要想这样做，可以将下列代码放到你的C代码中并确保它在任何线程被创建之前被调用。

```
#include <Python.h>
 ...
 if (!PyEval_ThreadsInitialized()) {
   PyEval_InitThreads();
 }
 ...
```

对于任何调用Python对象或Python C API的C代码，确保你首先已经正确地获取和释放了GIL。 这可以用 `PyGILState_Ensure()` 和 `PyGILState_Release()` 来做到，如下所示：

```
...
/* Make sure we own the GIL */
PyGILState_STATE state = PyGILState_Ensure();

/* Use functions in the interpreter */
...
/* Restore previous GIL state and return */
PyGILState_Release(state);
...
```

每次调用 `PyGILState_Ensure()` 都要相应的调用 `PyGILState_Release()` .

讨论¶

在涉及到C和Python的高级程序中，很多事情一起做是很常见的——可能是对C、Python、C线程、Python线程的混合使用。 只要你确保解释器被正确的初始化，并且涉及到解释器的C代码执行了正确的GIL管理，应该没什么问题。

要注意的是调用 `PyGILState_Ensure()` 并不会立刻抢占或中断解释器。 如果有其他代码正在执行，这个函数被中断知道那个执行代码释放掉GIL。 在内部，解释器会执行周期性的线程切换，因此如果其他线程在执行， 调用者最终还是可以运行的（尽管可能要先等一会）。