20.6. I am using a separate thread to run my code, but the application (or the UI) hangs.

There are a couple of hitches you can run into when trying to use threading and PyGTK together. For starters, if you are using threads, no matter if you are doing PyGTK calls from a separate thread or not, you *must* compile PyGTK with --enable-threads.

Now there are two approaches for threads in PyGTK:

1. Allow only the main thread to touch the GUI (gtk) part, while letting other threads do background work. For this to work, first call

```
gobject.threads_init()
```

at applicaiton initialization. Then you launch your threads normally, but make sure the threads never do any GUI tasks directly. Instead, you use gobject.idle_add to schedule GUI task to executed in the main thread. Example:

```
import threading
import time
import gobject
import gtk

gobject.threads_init()

class MyThread(threading.Thread):
    def __init__(self, label):
        super(MyThread, self).__init__()
        self.label = label
        self.quit = False

    def update_label(self, counter):
        self.label.set_text("Counter: %i" % counter)
        return False

    def run(self):
        counter = 0
        while not self.quit:
            counter += 1
            gobject.idle_add(self.update_label, counter)
            time.sleep(0.1)

w = gtk.Window()
l = gtk.Label()
w.add(l)
w.show_all()
```

```
w.connect("destroy", lambda _: gtk.main_quit())
t = MyThread(1)
t.start()

gtk.main()
t.quit = True
```

2. Allow any thread to do GUI stuff. Warning: people doing win32 pygtk programming have said that having non-main threads doing GUI stuff in win32 doesn't work. So this programming style is really not recommended.

Anyway, to make this work, start by calling:

```
gtk.gdk.threads_init()
```

at startup. Failing to do this will make PyGTK never release the python threading lock. At least Debian's packages are compiled properly, so it's a matter of using that call.

Then you have to wrap your main loop with gtk.threads_enter()/gtk.threads_leave(), like this:

```
gtk.threads_enter()
gtk.main()
gtk.threads_leave()
```

Your threads code must, before touching any gtk functions or widgets, call gtk.threads_enter(), and after gtk.threads_leave(), for example:

```
 ...
 gtk.threads_enter()
 try:
     myentry.set_text("foo")
 finally:
     gtk.threads_leave()
 ...
```

Also, keep in mind that signal handlers don't need gtk.threads_enter/leave(). There are other concerns, see [developer.gnome.org] .

Cedric Gustin posted a short example of threaded code at [www.daa.com.au] -- it's a good building block for a more complex threaded application.

Finally, if you are writing a C extension module, remember that you need to protect calls that potentially lock the thread with Py_BEGIN_ALLOW_THREADS and Py_END_ALLOW_THREADS.

Edit this entry / Log info / Last changed on Mon Aug 29 14:48:07 2005 by *Gustavo Carneiro (gjc@inescporto.pt)*