
Subject: Adding a socket to a GUI application

Posted by [Giorgio](#) on Mon, 22 May 2017 08:15:56 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi there,

I have my application with its GUI that connects to a db and makes stuff. It goes ok. Now I need the ability for the application to listen to a port and, when it receives a specific command, the application needs to modify its behaviour.

My main.cpp is like this

```
#include <CtrlLib/CtrlLib.h>
#include <Sql/sch_schema.h>
#include <Sql/sch_source.h>
//Other includes

using namespace Upp;

GUI_APP_MAIN
{
    String User, Pass, Schema, IP;
    int port;

    User = "foo";
    Pass = "bar";
    Schema = "test";
    IP = "192.168.1.2";
    port = 3306;

    //Connection to the DB
    MySqlConnection session;
    if(err_conn = session.Connect(User, Pass, Schema, IP, port)) {
        SQL = session;
        SqlSchema sch(MY_SQL);
        All_Tables(sch);
    }
    else {
        SetExitCode(1);
    }

    //Main window of the GUI
    HomeScreen hs;
    hs.Run();

}
```

I have no experience in sockets, but my guts tell me that I have to fork somewhere before the

.Run() command.

Any suggestion (including links to relevant documentation and RTFM) is appreciated.

Regards,

Gio

Subject: Re: Adding a socket to a GUI application
Posted by [nneilson](#) on Tue, 23 May 2017 05:06:47 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi
read the threads in the forum:
U++ MT-multithreading and servers.
and try a search in the upp forums for sockets.
Also do a google search on the internet for sockets.
Whether you have a gui app or not the operation of sockets is basically the same.
I use a socket to communicate from a upp c++ app with a Java app. If you stay within C++ it is a bit easier but the concept is the same. I think there is an example that comes with upp for the socket server and a client.

It will be interesting to see what you come up with for your db.

Neil

edit:

"I have to fork somewhere before the .Run() "

You may need to run the server code in a separate thread.

Subject: Re: Adding a socket to a GUI application
Posted by [Giorgio](#) on Wed, 26 Sep 2018 16:51:35 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi there,

I put this question aside for a while, few days ago I resumed it.

I had a look to threads and tcpsocket as suggested. For a starter I decided to focus on thread. So, I forked before the .Run() and put in the thread the code to manage the socket. I ended up with this:

```
void tagidSocket()
{
    //Socket
    RLOG("Socket's thread started");
```

```

for(;;) {
    if(Thread::IsShutdownThreads())
        return;
    RLOG("I'm still alive");
    Sleep(1000);
}
}

GUI_APP_MAIN{
    //Reading a .ini file, connection to a db
    [...]

    Thread t;
    t.Run(callback(tagidSocket));
    app.Run();

    RLOG("Exiting, terminating socket");
    Thread::ShutdownThreads();
}

```

This works as expected: I can use my application normally, I can see in the log the sentence "I'm still alive" several times, and when I close the applications it exits nicely / does not hung up.

After that I put the socket management in the equation and here came the problems. I began using the very same code used in the example "SocketServer". I copied everything in my tagidSocket() function. This is the result (GUI_APP_MAIN does not change):

```

void tagidSocket()
{
    TcpSocket server;
    if(!server.Listen(23456, 5)) {
        RLOG("Unable to initialize server socket");
        return;
    }
    RLOG("Socket started, waiting for requests...");
    for(;;) {
        if(Thread::IsShutdownThreads())
            return;
        TcpSocket s;
        if(s.Accept(server)) {
            String w = s.GetLine();
            //Cout() << "Request: " << w << " from: " << s.GetPeerAddr() << '\n';
            RLOG("Request: " + w + " from: " + s.GetPeerAddr() + '\n');
            if(w == "time")
                s.Put(AsString(GetSysTime()));
            else

```

```

    s.Put(AsString(3 * atoi(~w)));
    s.Put("\n");

}
}
}

```

With this change, when I launch my application everything is ok, I can connect to the socket and exchange data, but when I close my application it hangs and I have to kill it manually.

I try to debug the problem and I found out that the code responsible for the problem is the following: `if(s.Accept(server)) { [...] }`.

If I comment out everything in the if above (and also the if itself), the application can be closed normally (but of course a socket without the "listening" part makes no sense).

Why is this happening?

Regards,
gio

Subject: Re: Adding a socket to a GUI application
 Posted by [Oblivion](#) on Wed, 26 Sep 2018 20:03:59 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello Giorgio,

I'm afraid (as it'll make things somewhat complicated) what you seem to need is a socket in non-blocking mode.

Yet, there might be a simple solution for the test code you've provided:

```

void tagidSocket()
{
    TcpSocket server;
    if(!server.Listen(23456, 5)) {
        RLOG("Unable to initialize server socket");
        return;
    }
    RLOG("Socket started, waiting for requests...");
    try {
        while(!Thread::IsShutdownThreads()) {
            TcpSocket s;
            s.WhenWait = [=]
            {

```

```

    if(Thread::IsShutdownThreads())
        throw Exc("Thread is shut down.");
};
if(s.Accept(server)) {
    String w = s.GetLine();
    RLOG("Request: " + w + " from: " + s.GetPeerAddr() + '\n');
    if(w == "time")
        s.Put(AsString(GetSysTime()));
    else
        s.Put(AsString(3 * atoi(~w)));
    s.Put("\n");

}
}
}
catch(const Exc& e) {
    RLOG(e);
}
}

```

Now, the above code should work. But I can't guarantee it will continue to work in a complex code. That's why you need to get yourself familiar with non-blocking operations.

Best regards,
Oblivion

Subject: Re: Adding a socket to a GUI application
 Posted by [Giorgio](#) on Thu, 27 Sep 2018 07:23:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Oblivion,
 thank you for your support, I will read some documentation on blocking and non-blocking operations.
 Regards,
 gio
