Subject: Re: Simple IPC using Sockets
Posted by tojocky on Mon, 15 Feb 2010 13:43:17 GMT
View Forum Message <> Reply to Message

Nice,

I will test!
dolik.rce wrote on Mon, 15 February 2010 01:36Hello All,

I would like to share one very small class I wrote. It is simple object providing mostly automatic and very easy to set up interprocess communication. Example of usage is attached as well.

Basic goals: Simple to use
No specialized server app required
Everything should be as automatic as possible
Programmer can simply define communication protocol
Communication is done via Sockets, no shared memory (related to the next point)
It is not meant to be robust, but to allow easy and rapid development for small scale apps.
Implemented features: Everything mentioned in goals (I hope )
Protocol is specified using Callbacks, so rules can have different forms (e.g: single function with switch for handling messages OR several functions, each is defining one rule (used in example app) OR even pipeline style processing, where each function modifies the output of previous function and then passes it to next)
One instance of application performs the tasks of server in separate thread. This role is automatically picked up by another instance when the previous server is closed.
Optionally, programmer can start/stop server thread.

Warning: This package was not extensively tested yet, you might experience all sort of bad things from loss of information to deadlocks. It did not happen to me, but it doesn't mean it can't happen There is a high probability that I will develop this code further in future, since I wan't to use it in one of my projects.

About the example app: It's a console app that does nothing else then manages a list of strings (Vector<String> internally) from a simple shell. You can display it's content (command "show"), add ("add something") and delete items ("del number"). To exit, type "quit". Nothing much, heh? The interesting part is that if you open few more instances of this application, they will all behave as if they worked with single variable (but they of course do not, they just keep their own variable in sync). And this magic is done in ~50 lines of code, out of which is ~4O lines are definitions of protocol.

If you think this is interesting/useful please let me know. Suggestions for further development are welcomed too

Best regards,
Honza