Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 28 May 2020 09:36:26 GMT
View Forum Message <> Reply to Message

Hello slashupp,

Quote:I want to test if Terminal fits the needs of my app, but fail to find
a HowTo on installing it in my upp-dev-environment (which is standard)

Yes, this is missing from the docs. I will update it next week. In the meantime,

How to get it:

Terminal is a regular ctrl. It does not require anything other than U++ (ver >= 2020.1, because I use some new color methods which simplified the color management code.)

If you want to try it out, you can:

a) Clone the upp-components git repository (This is the recommended way to get updates,etc.) and set up an assembly in TheIDE for upp-components/CtrlLib (It is no different than adding another "bazaar")

b) Simply download the repo, unzip it, and move the CtrlLib/Terminal folder to your preferred folder.

Now that you have the source code, It should compile out of the box. There should be no need for tweaking, etc.

How to use it:

Well, the upp-components/Examples section already contains example code for different, basic use cases, I suggest you check there and play with them first (if you haven't already).
They are pretty simple. A minimal example is here upp-components/Examples/TerminalExample

(That minimal example, by the way, can run almost anything, out of the box)

Steps to write your own code:

1) Check the API docs first. Terminal ctrl contains regular topic++ files for TheIDE. It should be available through TheIDE's help system.

2) Create a CtrlLib application (basic or with layout, it's up to you. Terminal ctrl contains a layout file, so you can use it in TheIDE's layout editor.)

3) Add Terminal ctrl to you app. As it is a regular ctrl, it can be added to your window or any other "parent" ctrl via Ctrl::Add() method.

4) If you are going to use it as a local terminal, using a pty (currently on POSIX only), then Terminal package contains a PtyProcess class.
Add #include <Terminal/PtyProcess.h> to your app. Then all you need to do is set up the events and loop.
Provided examples use a rudimentary loop which is sufficient for most cases but you can write much efficient loops that'll suit your personal need and use-case.

5) If you are going to use it as a GUI front end for, say, SSH shells, add uppsrc/Core/SSH package to you app, and connect the loop. See the rudimentary upp-components/Examples/SshTerminalExample

Again, this is just generic information on how to install and use it.

If you have any specific questions in the meanitme, I'll be happy to address them. :)

Best regards,
Oblivion