Subject: A terminal emulator widget for U++
Posted by Oblivion on Sat, 23 Mar 2019 21:31:25 GMT
View Forum Message <> Reply to Message

Hello,

tl;dr: I am going to publish a terminal emulator widget for U++, in the following month (April).

From the user POV, it is a very simple Ctrl that can be embedded in any U++ applications via traditional U++ way: (E.g. Add(terminal.SizePos())
And it has a very "thin" public api.
Also its page and parser components are completely decoupled, and documented, and can be used seperately in any project (Read: You can write your own terminal emulator that suits your need with ease).

Before I publish it, I'd like to know what features you would like to see in it.

Currently it is more capable than Win10's new ANSI console.

At the moment it supports:

- VT52/ANSI/1xxx/2xx and -partially- 4xx/5xx emulation and Xterm extension.
- Both 7-Bit and 8-Bit modes.
- UTF8
- OCS and DCS
- ANSI colors
- Alternate screen buffer.
- Resize

Planned:

- Copy-paste support
- Scrollback buffer
- Mouse support
- And various optimizations.

I will also publish two reference examples with it: a simple terminal emulator constructed with Terminal widget, and a remote terminal to demonstrate SSH integration.

A screenshot is much more meaningful than words.
(On a humble test setup (Linux 5.0/Gnone/relatively old test AMD machine. Top row: Nano editor, AnsiArt, Vttest. Bottom row: Emacs, Top, RadeonTop).

Best regards.
Oblivion

File Attachments

1) Terminal Emulator.png, downloaded 2234 times

Subject: Re: A terminal emulator widget for U++
Posted by koldo on Sun, 24 Mar 2019 15:15:51 GMT
View Forum Message <> Reply to Message

Maybe a sample of it could be a kind of new Putty.
Putty is so basic that with any feature you would add, yours would be better. And in U++ :)

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 26 Mar 2019 09:02:38 GMT
View Forum Message <> Reply to Message

Hello Koldo,

Quote:Maybe a sample of it could be a kind of new Putty.
Putty is so basic that with any feature you would add, yours would be better. And in U++

Sure. Currently a barebone SSH terminal example (with the upconing Terminal ctrl), and TERM variable set to "xterm") in U++ is 52 lines of code. (Single file, password authentication).

What I have also in my mind is VirtualGui integration. :)

Think about accessing to your server or machine via a terminal running on web browser with SSL, on-demand.

This would be a very cool and (somewhat unique?) feature. And it's high on my TODO list. :)

Best regars,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 26 Mar 2019 12:11:12 GMT

View Forum Message <> Reply to Message

And here it is. It took five minutes. We now have terminal access over web browsers, using TURTLE :)
It works as expected.

This is the problem with U++: It is so good yet so underrated. :/

Screenshot:

Best regards,
Oblivion

File Attachments
1) 2019-03-26 15-06-40 ekran görüntüsü.png , downloaded 2082 times

Subject: Re: A terminal emulator widget for U++
Posted by koldo on Tue, 26 Mar 2019 12:21:22 GMT
View Forum Message <> Reply to Message

:lol:

Subject: Re: A terminal emulator widget for U++
Posted by Novo on Wed, 27 Mar 2019 22:14:19 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Tue, 26 March 2019 08:11And here it is. It took five minutes. We now have terminal access over web browsers, using TURTLE :)
It works as expected.

Are you able to run a Release version of your terminal emulator using TURTLE?
I'm asking this because I'm unable to run a Release version of WebWord (a demo app).

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 28 Mar 2019 07:01:33 GMT
View Forum Message <> Reply to Message

Hello Novo,

Quote:

Are you able to run a Release version of your terminal emulator using TURTLE?
I'm asking this because I'm unable to run a Release version of WebWord (a demo app).

Yep, it works on Linux on localhost:8088 with latest versions of U++, gcc. I have yet to test it (Turtle build) on CLANG or Windows.

Setting the Ctrl::host to localhost was sufficient.

The only problem I encountered was the loss of caret due to losing focus.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Novo on Thu, 28 Mar 2019 15:57:59 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Thu, 28 March 2019 03:01
Setting the Ctrl::host to localhost was sufficient.

Thanks a lot!

I didn't realize that Ctrl::host is used to connect back from JavaScript to a server.
I still have several problems like I cannot create more than one connection, although Release configuration is forking. And WebWord is using 100% of CPU because it is spinning in a connection loop which doesn't have any timeouts.

Are you experiencing the same problems or is it something related to my web-server setup?

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 28 Mar 2019 18:21:25 GMT
View Forum Message <> Reply to Message

Quote:
Are you experiencing the same problems or is it something related to my web-server setup?

Yes, after you've pointed out, I've checked the behaviour and it is indeed eating a lot of CPU cylcles (On Linux 5.0, GCC, release mode, AMD ryzen) I get 16% cpu usage in idle state).

Good news is I found a possible candidate and a workaround (or fix maybe?) but Mirek should see it first.

is waiting for connections) seems to be the problem:

A simple workaround or a possible fix is to call Sleep(10) in ln: 75.

E.g.

```
for(;;) {
 if(quit)
  return false;
 Sleep(10); // Let's eat less CPU cycle while waiting for connections.
//...
```

OTOH, there are also bad news:

1) It ssems impossible to open multiple clients at once (at leasy on Firefox (latest).
   Symptom: When trying to open a second connection to turtle server, tab immediately closes (yet a second webword process runs in the background!).
2) I've noticed a potential security flaw: Closing the client using the client's "close window" button does not clear the browser tab (or canvas). It should. Or else sensitive data may be visible.

Best regards.
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Mon, 08 Apr 2019 09:45:43 GMT
View Forum Message <> Reply to Message

As the day that I'll publish the code on my github repo draws closer, final bits for the first version of TerminalCtrl are coming together well.

- Scrollback buffer is implemented.
- Chameleon (theme) support is implemented.
- VT52/1xx/2xx keyboard (with function keys) support is implemented.
- Conformance levels support is added (i.e. it is now possible to restrict the operation level to e.g. VT52, VT102, or VT220 etc.)

Here is a screenshot.
On the left is our SSH package/SshShell (running in console mode) running on TerminalCTrl, with scrollback buffer enabled.
In the middle is TerminalCtrl running with the widely used "Solarized theme"
On the right is the mighty emacs, with dropdown menus and function keys on default theme.

Suggestions are welcome.

Best regards,
Oblivion

File Attachments

1) Chameleonized-TerminalCtrl.png, downloaded 1990 times

Subject: Re: A terminal emulator widget for U++
Posted by Xemuth on Mon, 13 May 2019 11:47:59 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

Very good job, it seems to be awesome.
I will try it on my Raspberry with TURTLE !

When do you think you'll publish it ? :blush:

Thanks in advance,
Best Regard

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 14 May 2019 07:57:03 GMT
View Forum Message <> Reply to Message

Hello Xemuth,
Xemuth wrote on Mon, 13 May 2019 14:47Hello Oblivion,

Very good job, it seems to be awesome.
I will try it on my Raspberry with TURTLE !

When do you think you'll publish it ? :blush:

Thanks in advance,
Best Regard

Terminal packaga is already delayed (initially I'd planned to publish it in April).
Good news is that the first version of the package is basiaclly complete. So it will be published at

the end of this month.

Currently I am making some profiling and optimizations (which led to 37% performance gain and 30% less memory consumption on average.
Also, now the parser is a complete vt parser, i.e. it can handle every single VT instruction (not every command is implemented though) correctly from VT52 to Vt520)
I am testing it with vttest, xterm's and ncurses's test tools. And fixing some "paper-cuts" that I have noticed. :)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 08 Jun 2019 14:48:00 GMT
View Forum Message <> Reply to Message

Hello,

A small but "big" update.

Terminal ctrl for U++ is on its way, and here is a short video preview:

https://vimeo.com/341085501

It is now mostly compatible with xterm.

-- Mouse tracking support added. Terminal can now handle mouse events if they are supported by the applications (See above video for example)
-- Bracketed paste mode added.
-- "G-sets" upport added. It now supports legacy applications that rely on font shifting (g0, g1, g2, g3, gl, gr).
-- DEC Technical charset is added.

Now I feel it is eligible for first release. (Probably next friday...)

P.s.: Terminal may seem sluggish in the video. (It is not.)
That's because I encoded it on my good old test machine, and I limited the framerate to 15.

Best regards,
Oblivion.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 13 Jul 2019 15:35:03 GMT
View Forum Message <> Reply to Message

Hello,

It's been a while since I posted any news about the Terminal package. I had to focus on my other works fo a while.

The good news is last week I implemented the final missing pieces for the 0.1 release:

- 256 colors support added.
- ANSI colors support added.
- XTerm dynamic colors suppport added.
- VT 4xx rectangular area operations are added (copy, invert, move, fill, in both selective and normal modes)
- UDK (DEC's user-defined function keys support) added.
- Lazy resize option is added (to reduce flickers on network terminals uch as SSH-based ones)
- Size hint added.
- A .usc file is added to the Terminal package. (The most common options (font, ink, paper, cursor, sizehint etc.) can be set using TheIDE's layout editor. Also it shows a size hint (in calculated cell size) to simplfy positiong the widget in the layout ediyor.
- It is also tested on Windows, and it works well. :) (currently as SSH terminal, in the near future as a frontend for Windows power shell too)

Two notes on the upcoming initial release:

1) Terminal package currently does not contain any external code/libaray. It uses U++, and it's plugins. :)
2) Although a virtual terminal requires a pty device, and Terminal pacjage contains one, they are completely decoupled.
   Terminal ctrl can be used and compiled without PtyProcess. This gives it a huge flexibility
   In this regard I will provide 4 basic examples with the package:
   - TerminalExample | Uses ptyprocess (currently ptyprocess requires POSIX-compliant operationg systems (or possibly cygwin on Windows.)
   - TerminalExampleWithLayout | The same as above.

   - SShTerminalExample | Does not use PtyProcess. IT uses Core/SSH package isntead
   - SshTerminalExampleWithLayout | The same as above.

Here is the actual code of TerminalExample (36 LOCs total):

```
#include <Core/Core.h>
#include <Terminal/Terminal.h>
```

```
using namespace Upp;

const char *nixshell = "/bin/bash";

struct TerminalExample : TopWindow {
 Terminal term;
 PtyProcess pty;

 TerminalExample()
 {
  term.WhenBell =   [=]()  { BeepExclamation(); };
  term.WhenTitle =   [=](String s) { Title(s); };
  term.WhenResize =   [=]()  { pty.SetSize(term.GetPageSize()); };
  term.WhenOutput =   [=](String s) { PutGet(s); };
  SetRect(term.GetStdSize()); // 80 x 24 cells (scaled).
  Sizeable().Zoomable().CenterScreen().Add(term.SizePos());
  SetTimeCallback(-1, [=]()  { PutGet(); });
  pty.Start(nixshell, Environment(), GetHomeDirectory());
 }

 void PutGet(String out = Null)
 {
  term.Write(pty.Get());
  pty.Write(out);
  if(!pty.IsRunning())
   Break();
 }
};

GUI_APP_MAIN
{
 TerminalExample().Run();
}
```

Below was a sort of "final boss" for the first release. It shows the mapscii, an OpenStreetMap implementation for terminal devices, running on the above code and on Gnome-terminal.
On the left is TerminalExample, running mapscii.
On the right is gnome terminal running mapscii
Both are running on 256 colors mode + mouse tracking support. :)

As a final note: Terminal package will be availabe within this weeek.

Best regards,
Oblivion

File Attachments

1) Terminal.png, downloaded 1684 times

Subject: Re: A terminal emulator widget for U++
Posted by Xemuth on Tue, 16 Jul 2019 07:34:35 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

Impressive work !

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 20 Jul 2019 13:33:51 GMT
View Forum Message <> Reply to Message

Hello,

It is time to unleash the beast :)

The initial version of Terminal package is finally released.

I suggest you read the ReadMe.md on GitHub or the "Overview" doc (qtf) inluded in the package. It'll give you a better idea of what it is.

Github address of Terminal package:
https://github.com/ismail-yilmaz/upp-components/tree/master/ CtrlLib/Terminal

Examples are included in the Examples section of my git repo.

Yes, but what is it?

Excerpt form the ReadMe.md on github:

Terminal package is a flexible, easy-to-use yet powerful cross-platform virtual terminal emulation library written in C/C++ for Ultimate++.

It is designed from the ground up with modularity and maintainability in mind.
In this respect the package consists of several re-usable classes, only one being the Terminal

widget.

Requirements

- Ultimate++ (ver. >= 2019.1)
- POSIX, Windows (and probably MacOS, though not tested.)
- A decent enough C/C++ compiler that supports at least C++11. (GCC/CLANG/MinGW/Msc)
- Snacks & beer.

Features

*Note that below list is only a summary of the currently supported features. Technical specifications and documentation will be available soon.

- Supports whatever platform Ultimate++ supports. (Linux, Windows, MacOS).
- Supports VT52/VT1xx/VT2xx, partial VT4XX/5XX, and xterm emulation modes.
- Supports user configurable device conformance levels (1, 2, 3, 4, and 0 as VT52 emulation).
- Supports both 7-bits and 8-bits I/O.
- Supports Unicode/UTF8.
- Supports user configurable, legacy "g-set" (G0/G1/G2/G3), and related shifting functions (LS0/LS1/LS1R/LS2/LS2R/LS3/LS3R).
- Supports ANSI conformance levels.
- Supports various terminal state, device, and mode reports.
- Supports DEC VT52 graphics charset, VT1xx line-drawing charset, VT2xx multinational charset, and VT3xx technical charset.
- Supports VT52/VT1xx/VT2xx keyboard emulation with function keys.
- Supports UDK (DEC's user-defined function keys feature).
- Supports user configurable blinking text and blink interval.
- Supports ANSI colors (16 colors palette).
- Supports ISO colors (256 colors palette).
- Supports xterm dynamic colors (dynamic ink/paper/selection colors).
- Supports bright colors.
- Supports background color erase (BCE).
- Supports transparency (i.e. allows background images).
- Supports VT4xx rectangular area operations: copy, invert, fill. erase.
- Supports VT4xx rectangular area checksum calculation and reporting.
- Supports both DEC and ANSI style selective erases.
- Supports alternate screen buffer.
- Supports history/scrollback buffer.
- Has a user switchable scrollbar.
- Supports xterm style alternate scroll.
- Supports resize (and optional lazy resize to reduce flicker on network terminals such as SSH-based ones).
- Supports both immediate display refresh and delayed (buffered) display refresh.
- Supports xterm style mouse tracking: button, wheel, motion, focus in/out events.
- Supports user configurable cursor styles (block, beam, underscore, blinking/steady).
- Supports cursor locking.
- Supports clipboard operations (copy/paste/select all - including history buffer) and basic

drag-and-drop ops.
- Supports bracketed paste mode.
- Has a predefined yet programmable context menu (left mouse button menu).
- Supports window titles.
- Supports bell notifications.
- Supports VT1xx LEDs.
- Supports size hint.
- Supports Ultimate++ style data serialization.
- Supports per-widget customization (i.e no global variables or properties are used).
- Includes a Terminal.usc file for TheIDE's layout editor.

Reviews, bug reports, patches, suggestions are welcome.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 21 Jul 2019 19:15:20 GMT
View Forum Message <> Reply to Message

Hello,

A Turtle example is added to the package.
You can run the Terminal example in your favorite browser and see how it works. ;)

Terminal: https://github.com/ismail-yilmaz/upp-components/tree/master/ CtrlLib/Terminal
Example: https://github.com/ismail-yilmaz/upp-components/tree/master/ Examples/TerminalInWebBrowser

Also a name clash with Turtle is fixed.

Screenshot: Lynx running on Terminal in Firefox (a.k.a "Browserception")

Best regards,
Oblivion

File Attachments
1) TerminalExample-Turtle.png, downloaded 1760 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Mon, 22 Jul 2019 13:56:03 GMT
View Forum Message <> Reply to Message

Hello,

More updates:

A terminal multiplexing example is added to the package.

This example demonstrates how a terminal multiplexing can be achieved simply by using a splitter widget. Splitter is a container ctrl that can be used to split any parent ctrl into resizeable horizontal and/or vertical panes. It also demonstrates the usage of NTL containers with Terminal widget:

Also a mouse capture issue is hopefully fixed.

Best regards,
Oblivon

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Wed, 24 Jul 2019 11:24:51 GMT
View Forum Message <> Reply to Message

A short video demonstration of Terminal ctrl (simple multiplexing example running mapscii, htop, emacs, lynx nano..):

https://vimeo.com/349761874

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by koldo on Fri, 26 Jul 2019 06:18:07 GMT
View Forum Message <> Reply to Message

Terminal based world map viewer is cool :)

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 27 Jul 2019 09:30:37 GMT
View Forum Message <> Reply to Message

Hello Koldo,

Yeah it's very impressive tool. I enjoyed it.
It also helps a lot for testing.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Mon, 29 Jul 2019 21:23:38 GMT
View Forum Message <> Reply to Message

Hello,

A small but important update.

True/direct (24-bit) color support has landed.

It is added as a compile-time option this time. (It can be easily switched using TheIDE)
The reason for this decision is that true color support increases the size of each cell by 4 bytes.
(Not worriyingly high but I'm being cautious. :))

Latest code of terminal package can be found at:
https://github.com/ismail-yilmaz/upp-components/tree/master/ CtrlLib/Terminal

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 02 Aug 2019 11:54:12 GMT
View Forum Message <> Reply to Message

Latest round of updates:

Terminal: Clipboard and DND operations improved.
Terminal: Drag animation for drag-copy operation implemented.
Terminal: WhenClip event added. (Allows client code to inspect/accept/reject pasted/dropped clips.
Terminal: Api doc updated accordingly.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Wed, 07 Aug 2019 19:24:49 GMT
View Forum Message <> Reply to Message

Hello,

Terminal package is updated.
Initial support for a rare but much requested feature in the world of terminal emulators has landed:

- Sixel graphics support is added to the Terminal package. It is now possible to view sixel images, using the Terminal package. However, the initial support is only for external viewing. Embedded images are a TODO.

- A basic terminal example with sixel viewer is also added to the package.

- SixelRenderer class and a convenience function: RenderSixelImage is also added to the package. This class and the function can also be used as a stand-alone sixel renderer.

- Alt-key handling is improved.

A screenshot:

The git repo address of upp-components: https://github.com/ismail-yilmaz/upp-components

Reviews, criticism, patches, bug reports, etc., are always welcome.

Best regards,
Oblivion

File Attachments
1) Terminal-SixelViewer.jpg, downloaded 1383 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Mon, 12 Aug 2019 09:39:52 GMT

Hello,

A small update on the progress of the Terminal ctrl's embedded image (sixel, etc.) support. 8) (Note that I have yet to push changes. I'll problably push the changes next week, as there are some minor issues to solve.)

This will not break the existing behavior. Sixel support will be available in both embedded and external mode.

Best regards,
Oblivion

File Attachments

1) Terminal-Embedded-Sixel.jpg, downloaded 1275 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 16 Aug 2019 12:29:25 GMT

Hello,

A day for big updates :)

Terminal ctrl has gained embedded images and Upp::Display support.

Terminal: Embedded images support is added (currently, sixel).
Terminal: Renderer is improved and further optimized.
Terminal: Upp::Display support for image objects are added.
Terminal: LeftAlignedImageDisplay() and RightAlingnedImageDisplay() functions added.

Screenshot:

You can always find the new version here:

https://github.com/ismail-yilmaz/upp-components/tree/master/ CtrlLib/Terminal

Best regards,
Oblivion

File Attachments
1) Terminal-Screenshot.jpg, downloaded 1252 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 22 Sep 2019 10:01:47 GMT
View Forum Message <> Reply to Message

Hello,

Terminal package is finally updated to v0.2.

Image rendering and handling mechanism is vastly improved. :)

- Terminal: SixelRenderer class is rewritten;
- Terminal: SixelRaster class, a StreamRaster interface for sixel images, is added to the package.
- Terminal: A subset of xterm's window-op reports are implemented.
- Terminal: An I/O synchronization issue with the PtyProcess class that led to artifacts on screen on some setups is fixed.
- Terminal: Inline image support is improved.
- Terminal: Room is made for other inline image protocols.
- Terminal: WhenSixel event is removed in favor of a generic WhenImage event.
- Terminal: Image rendering strategy is changed to cell-level rendering for further flexibility.
- Terminal: An LRU-type shared image data cache is implemented.
- Terminal: Support for Jexer, a modern text user interface (TUI) and windowing system for terminal emulators, is added.
- Terminal: A general purpose data variable is added to the cell structure.
- Terminal: SGR image flag is added to the cell structure.
- Examples: Examples are further simplified.
- Various optimizations and fixes.
- Image gallery is updated.

The "final boss" for this release was the jexer support. Jexer is a modern text user interface and window manager that runs inside terminal emulators. (It is an advanced terminal multiplexer.)

There were only four high-end terminal emulators (xterm, mlterm, rlogin and jexer itself) fully supporting the jexer, Now they are five :)

I've made short videos of Terminal ctrl.
The ssh terminal example on windows also demonstrates the power of Core/SSH package. (It connects to a Linux device over 300 km distance, and runs jexer with inline images support, which is very demanding). 8)

On Linux
- A basic terminal example with sixel graphics, and mouse tracking support.

- Used apps and tools: Jexer text user interface (TUI), GNUPlot, Emacs, Nano, htop, ncurses demos.
- Link: https://vimeo.com/359241367

On Windows
- A basic SSH2 terminal example with sixel graphics, and mouse tracking support.
- Used apps and tools: Jexer text user interface.
- Link: https://vimeo.com/361556973

On Turtle HTML-5 backend (in a web browser)
- A basic terminal example with sixel graphics, and mouse tracking support.
- Used apps and tools: Jexer text user interface.
- Link: https://vimeo.com/361558519

Screenshot (Windows, ssh terminal example):

Terminal@: https://github.com/ismail-yilmaz/upp-components/tree/master/ CtrlLib/Terminal

Best regards,
Oblivion

File Attachments

1) terminal_jexer_windows_screenshot.jpg, downloaded 1277 times

---

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 28 Sep 2019 20:00:38 GMT
View Forum Message <> Reply to Message

Hello,

A new example is added to the package: Ssh terminal splitter.

This example demonstrates the interaction between the Terminal widget and Ultimate++'s Core/SSH package, in a multithreaded environment.
Therefore it also demonstrates the usage and power of U++s own Core/SSH package in a multithreaded GUI environment.

Link to a short video showing the ssh terminal splitter in action on Windows:

https://vimeo.com/362532208

A screenshot taken from the above video:

Best regards,
Oblivion

File Attachments

1) terminal_ssh_splitter_windows.jpg, downloaded 1188 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 03 Nov 2019 14:03:14 GMT
View Forum Message <> Reply to Message

Hello,

Terminal package and Terminal ctrl is updated.

There are fixes and clean-ups here and there, but most notably a new feature is added:

Terminal ctrl is now able to display 24-bit jpeg, gif, bmp, etc. raster formats, using the Jexer Image Protocol and Upp's own StreamRaster interface.

This means that if the raster images you are working with have registered decoders in upp/plugins, or somewhere else, Terminal ctrl will be able to display it.

Four main benefits of this inline images protocol is:

1) It is "8-bit clean". It can be safely deliver images over the network, and with Utf-8.
2) Popular image formats can be decoded much faster than sixels. (Up to 8 to 10 times faster on bigger image files), and usually have a lower bandwidth requirements.
3) Allows 24 bit (true color) images.
4) The wire protocol is very simple.

Screenshot with 24 bit images:
(Taken from the pre-alpha version of Toad, a cross-platform, multithreaded SSH2 client with tabs and splitter support, that will be available in -hopefully- January 2020):

Best regards,
Oblivion

File Attachments
1) toad_jexer_ssh.png, downloaded 1214 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 10 Nov 2019 20:49:45 GMT
View Forum Message <> Reply to Message

Hello,

A new week, and another update with new features. :)

- Explicit hyperlinks support is added.
- Standard menu improved and modularized.
- Cell tracking methods are implemented.
- Various fixes and cosmetics.
- Docs are updated accordingly.

Explicit hyperlinks are currently supported by GCC, ls, systemd, and a number of other high profile products.
One advantage of this hyperlinks protocol is that it is relatively well-defined and cheap.

To test it on linux, compile the TerminalExample, and run an updated version of the "ls" command:

ls -l --hyperlink

Gif:

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 12 Nov 2019 15:36:23 GMT
View Forum Message <> Reply to Message

Hello,

I have pushed another update, which contains features that were brewing for a while:

- Terminal: Rectangle selection is now supported.
- Terminal: Iniline images can now be copied to clipboard or sent to client. (It uses the same mechanism as hyperlinks: CTRL + double click or context menu)
- Terminal: Options menu is implemented. Options menu contains the most relevant options for terminal.

Screenshot:

Best regards,
Oblivion

File Attachments
1) terminal_rectangle_selection._linux_screenshot.jpg, downloaded 1084 times

---

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 12 Nov 2019 19:08:40 GMT
View Forum Message <> Reply to Message

As a last minute bonus:

DnD operations and animations are added for both hyperlinks and inline images. (Activated with CTRL-key modifier)

:)

Gif:

Enjoy!

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Wed, 25 Mar 2020 13:20:10 GMT
View Forum Message <> Reply to Message

Hello,

Three months of silence is finally broken. :)

Terminal package is scheduled to be updated to v0.3 on April 6, 2020.

Meanwhile the package has seen tons of improvements. (Note that the changes are not pushed to git yet).

Some highlights of the upcoming version:

- Initial steps for scripting support (using the Esc language).
- A proper double-width (Eastern-Asian/CJK) characters support.
- A proper tmux/screen support.
- A proper support for ISO 8613-6 true/indexed color formats.
- A much faster renderer. (In some torture tests (such as ncurses/dots on fulscreen mode) the performance gain can be up to %80.

And much more... (full list will be available with the release).

I am currently testing the package for regressions.

To whet yout appetite (tmux/screen, on Linux):

Best regards,
Oblivion

File Attachments

1) terminal_tmux.png, downloaded 1065 times

Subject: Re: A terminal emulator widget for U++
Posted by deep on Wed, 25 Mar 2020 18:12:36 GMT
View Forum Message <> Reply to Message

Hi Oblivion

For double-width font support are you using Harfbuzz? Will it support Indic scripts?

Quote:- A proper double-width (Eastern-Asian/CJK) characters support.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 26 Mar 2020 15:54:03 GMT
View Forum Message <> Reply to Message

Hello Deepak,

Quote:Will it support Indic scripts?

This is my intention, with other scripts (Arabic, etc.) and Bi-directional rendering.

But unfortunately it will not happen in the upcoming v0.3 release.
At the moment what I can provide is double witdth support for CJK ideographs and other double width Asian characters.

As for the harfbuzz, currently I am using what U++ uses on GTK and/or X11.
And AFAIK, GTK 3.x uses harfbuzz by default.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 10 Apr 2020 00:17:43 GMT
View Forum Message <> Reply to Message

Hello,

The day has come. Terminal package is upgraded to v0.3. :)

This is a huge release, with lots of changes, fixes, additions and improvements.

I am not going to list every change but some highlights here. If you want to see the full list, please check out the git history.

- Source code is restructured.
- Console class is ditched for the sake of simplicity. There is now only Terminal, deriving from Upp:CTrl.
- Eastern-Asian/CJK double width characters support is added.
- Text overline attribute is now supported.
- APCs (application programming commands) are now supported. This is the first step towards terminal scripting feature. APCs can be also be used by client code for private purposes.

- TRUECOLOR flag is removed. Terminal ctrl is now fully and officially a true color terminal emulator.
- Accordingly, CMY and CMYK color spaces are now supported too!
- Support for color text specifications (rgb, hash3, hash6, hash9, hash12) are added.
- xterm window actions are reports are implemented. It is now possible for host apps to move, resize, minimize, maximize, fullscreen the terminal.
- Renderer has seen a significat performance boost (up to 80% on some torture tests).
- tmux/screen is fully supported.
- Hyperlinks now show nice dots under their texts when they are "inactive" (i.e. no mouse hovering over them)
- API docs and specifications docs are significantly improved.

But how compliant is it?

Using George Nachmann's (iTerm2's developer) excellent and comprehensive test suite (it is analogous to the browser standard compliance tests), esctest, the results are as follows:

expected-terminal=xterm
max-vt-level=4

Terminal ctrl: v0.3, passed 407 tests, failed on 101 tests,

Terminal ctrl: v0.2 passed only 160 of the same tests.

Let us now look at the latest stable version of Gnome terminal and Kitty, with same settings:

expected-terminal=xterm
max-vt-level=4

Gnome Terminal (v.3.36.1.1): passed 163 tests and failed on 345 tests.

Kitty: passed 122 tests, and failed on 386 tests.

xterm: Fassed all tests. (not suprisingly)

Keep in mind that I am not even implying that Terminal ctrl is better. (It isn't. Gnome Terminal is a very high quality product with excellent features).

I am simply pointing out where our VT engine stands on the xterm compliance front (since xterm is the defacto standard), and the progress made with the nev version.

Hyperlinks with dots:

You can find the new version on my git repo.

If you have any questions, suggestions, bug reports, patches, feature requests, etc. please feel free to contact me via my github address or this forum topic.

Enjoy!

Best regards,
Oblivion

File Attachments

1) hyperlink_dots.png, downloaded 1022 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 10 Apr 2020 22:30:11 GMT
View Forum Message <> Reply to Message

A last minute bonus:

Terminal ctrl now supports the popular iTerm2's inline image protocol too. :)
(See api and specification docs for more details)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 17 Apr 2020 20:30:12 GMT
View Forum Message <> Reply to Message

Hello,

Terminal ctrl has received a small but important update:

Autowrap mode is fixed.

The mode setting somehow got lost in the v0.2 -> v0.3 transition. :blush:

I recommend you to update the package as the autowrap feature is widely used by command line applications.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 21 Apr 2020 16:35:14 GMT
View Forum Message <> Reply to Message

Hello,

Terminal has gained two small features.

- Jsonization support is added.
- Xmlization support is added.

It is now possible to store/load the configuration (including color table) of Terminal ctrl instances in/from xml and json files, in a human readable form.

Also, missing tester methods (i.e, Isxxx()/Hasxxx()) are added and docs are updated accordingly.

Best regards,
Oblivion.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Wed, 22 Apr 2020 16:51:52 GMT
View Forum Message <> Reply to Message

A small but important news is that I've found a bottleneck in the Terminal ctrl's code that was puzzling me for some time.
It was in the display update and scrolling mechanism. Hopefully I have fixed it. :)

However, I won't push the change to github before I test it thoroughly and be sure it doesn't break anything.

It appears that, after the fix, Terminal ctrl will be one of the most fastest out there.

Below statistics are the average results of a test automatically run 100 times:

All terminal emulators use the same font/page size settings, and run on Gnome 3.36

Results of command: time find /usr/share

Terminal Emulator	real	user	sys
Terminal Ctrl (Current):	0m23,104s	0m1,297s	0m2,879s
Terminal Ctrl (with fix):	**0m4,002s	0m0,774s	0m1,622s**
xterm	0m15,059s	0m0,991s	0m2,527s
Gnome Terminal	0m4,217s	0m0,778s	0m1,773s
Kitty	0m4,907s	0m0,887s	0m2,325s

If everything goes well, I will push this change next week.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 24 Apr 2020 22:36:07 GMT
View Forum Message <> Reply to Message

Hello,

I have pushed the the critical performance fix that I mentioned in my last message.

The main culprit was using a Vector where a BiVector was a much better option. In fact, it seems to be the optimal solution to the scrollback/hsitory buffer problem. Removing elements from the beginning of a Vector, especially when its length > 64K is -not surprisingly- very slow. BiVector solves exactly this problem because a scrollback buffer is mutated only from the either end.

There was also a constant display refresh happening when a page was scrolled, even when a single line scrolled, degrading performance. This is fixed too.

Now Terminal ctrl's scroll speed / throughput can be very fast. (It is Even faster than the above given numbers: ~3.5 secoonds on the same test while acting as a both SSH (using Upp/SSH package) and local terminal. ("time find /usr/share" command was outputting > 300.000 file paths in these tests, with the size of scrollback buffer of each terminal was set to 64 k)

However, you can't see the real difference if you use the provided reference example codes, because they have a very rudimentary event loop.

If you are curious and you want to test the scrolling/throughput performance of Terminal ctrl, you can use the below code: While still rudimentary, it is closer to real world scenarios.

```
#include <Terminal/Terminal.h>
#include <Terminal/PtyProcess.h>
```

```
using namespace Upp;

const char *nixshell = "/bin/bash";

struct TerminalExample : TopWindow {
 Terminal term;
 PtyProcess pty;

 TerminalExample()
 {
  SetRect(term.GetStdSize());
  Sizeable().Zoomable().CenterScreen().Add(term.SizePos());
  term.WhenBell   = [=]()               { BeepExclamation(); };
  term.WhenTitle  = [=](String s)       { Title(s);          };
  term.WhenOutput = [=](String s)       { pty.Write(s);      };
  term.WhenResize = [=]()               { pty.SetSize(term.GetPageSize()); };
  term.WhenLink   = [=](const String& s) { PromptOK(s);      };
  term.InlineImages().Hyperlinks().WindowOps().DynamicColors();
  term.SetHistorySize(65536);
 }

 void Run()
 {
  pty.Start(nixshell, Environment(), GetHomeDirectory());
  OpenMain();
  while(IsOpen() && pty.IsRunning()) {
   int ms = 16;
   String in = pty.Get();
   if(!IsNull(in)) {
    term.WriteUtf8(in);
    int len = in.GetLength();
    if(len >= 1024)
     ms = 1024 * 16 / len; // Scale to workload...
   }
   ProcessEvents();
   Sleep(ms);
  }
 }

};

GUI_APP_MAIN
{
 TerminalExample().Run();
}
```

There are other improvements and bug fixes too (mostly on serialization/xmlization/jsonization fronts).

If you have any questions, criticism, bug reports, patches, ideas, let me know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 15 May 2020 16:01:19 GMT
View Forum Message <> Reply to Message

Hello,

Terminal ctrl is updated. It has gained a new feature that is found on most popular vt emulators:

- Auto-hide option for mouse cursor is added. Terminal ctrl can be configured to automatically hide the mouse pointer when the user types a text.

There are also small bug fixes here and there. :)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 17 May 2020 20:20:55 GMT
View Forum Message <> Reply to Message

Hello,

Ranger, a terminal based file manager, can now show image previews via iterm2's inline image protocol, on terminal ctrl.

By the way, I am planning to implement optional, range-based font substitution for Terminal ctrl v0.4 (2020.2, synced with U++).

My initial plan is to give the client the ability to compose a font list (using a simple structure called Terminal::RangeFont), and pass it to terminal ctrl instances so the client will be responsible for taking care of the list. (Similar to how Upp::Display objects are handled.)

If you have any other suggestions I have opened a ticket for it. Feel free to share your ideas.

Best regards,
Oblivion.

File Attachments
1) termina_ranger_linux_screenshot.png, downloaded 1158 times

---

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 22 May 2020 14:08:08 GMT
View Forum Message <> Reply to Message

Hello,

This week on Terminal ctrl: :)

- Initial support for copy-on-select style (X11-style) text selection is added.
- Terminal ctrl recognizes CMY/CMYK and urxvt-style RGBA color text specifications via dynamic colors extension.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by slashupp on Thu, 28 May 2020 06:30:48 GMT
View Forum Message <> Reply to Message

Hi oblivion

Just found this - looks great!

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)
in order to use/test using it in my app.

Can you pls supply a HowTo for access/use of Terminal in a blank/new project?

thx

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

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 29 May 2020 10:40:48 GMT
View Forum Message <> Reply to Message

An example demonstrating the absolute minimum setup for a standalone terminal app, and two different loops:

```
#include <Terminal/Terminal.h>
#include <Terminal/PtyProcess.h>

using namespace Upp;
```

```
struct MinimalTerminalExample : TopWindow {
    Terminal term;          // A terminal ctrl instance.
    PtyProcess pty;         // A pseudoconsole process instance.
    MinimalTerminalExample()
    {
        SetRect(0, 0, 800, 600);
        Sizeable().Zoomable().CenterScreen().Add(term.SizePos());
        term.WhenOutput = [=](String out)   { pty.Write(out);  };                    // Terminal output
        term.WhenResize = [=]               { pty.SetSize(term.GetPageSize()); };    // Notifies the pty
about page size changes.
        pty.Start("/bin/bash", Environment(), GetHomeDirectory());                   // Start a shell
process on pty.
        // For a reasonable throughput, a callback-based loop is sufficient.
        SetTimeCallback(-1, [=] { term.WriteUtf8(pty.Get()); if(!pty.IsRunning()) Break(); });
    }

//  For very high throughput, below loop (or something better) can be used.
//
//  void Run()
//  {
//      pty.Start("/bin/bash", Environment(), GetHomeDirectory());
//      OpenMain();
//      while(IsOpen() && pty.IsRunning()) {
//          ProcessEvents();
//          String s = pty.Get();   // PtyProcess reads are non-blocking.
//          if(!IsNull(s))
//              term.WriteUtf8(s);
//          int l = s.GetLength();
//          Sleep(l >= 1024 ? 1024 * 10 / l : 10); // Scale...
//      }
//  }

};

GUI_APP_MAIN
{
    MinimalTerminalExample().Run();
}
```

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 06 Jun 2020 11:57:44 GMT
View Forum Message <> Reply to Message

Hello,

Terminal ctrl is updated. It has a number of under-the-hood improvements that lead to higher performance/throughput.
To give you an idea, here are some benchmark results I've been compiling (both synthetic and real-usage) for some time:

CPU: AMD FX 6100 - Black Edition (Six-core processor)
Memory: 8 GiB, DDR-3 1800 Mhz
Storage: 128 Gib SSD, 1Tib HDD
Graphics: AMD Radeon R7 240, 2GiB
Monitor: LCD, HD (1080p)
OS/Kernel: Arch/Linux 5.6 (x86_64)
Desktop: GNOME 3.36 (X11)
Framework: Ultimate++ 2020.1 (alpha)
Flags: GUI
Compilers: GCC 10.0.1 CLANG 10.0.0
Application: TerminalExample (with a high throughput loop).
Font: The same IBM Plex Mono, monospace font used in all tests.

Disclaimer: The benchmarks below do not represent any competition.
They don't represent any weak spots of the mentioned terminals. They are all high quality stuff.
The sole purpose of these benchmarks is to give an idea as to where the Terminal ctrl's "engine" performance stands ATM.

Note: All benchmarks are repeated 50 times, automatically.
The performance difference between GCC/CLANG builds is negligable.
The latest stable versions of the mentioned terminals are used in these tests (as of May-June 2020).

Results of command: "time find /usr", file count: 415897, page size: 237 x 55, Monospace Font)

Terminal Emulator	real	user	sys	Notes
Alacritty	0m3,776s	0m1,134s	0m2,637s	Scrollback buffer size: 64k
Terminal ctrl	0m4,126s	0m1,202s	0m2,466s	Scrollback buffer size: 64k
Kitty	0m5,608s	0m1,224s	0m2,763s	Scrollback buffer size: 64k
Gnome Terminal	0m6,785s	0m1,243s	0m3,015s	Scrollback buffer size: 64k
xterm	0m24,368s	0m1,661s	0m3,766s	Scrollback buffer size: 64k

Terminal Emulator	real	user	sys	Notes
Simple terminal (0.8)	0m3,791s	0m1,162s	0m2,533s	Has no scrollback buffer.

Terminal ctrl (0.3) 0m3,971s 0m1,120s 0m2,472s Scrollback disabled.

Terminal Emulator	real	user	sys	Notes
Terminal ctrl (0.3)	0m4,380s	0m1,479s	0m2,536s	localhost, Scrollback buffer size: 2000 lines, no compression, (SshTerminalExample is used)
PuTTY (0.73)	0m40,828s	0m1,354s	0m3,071s	localhost, Scrollback buffer size: 2000 lines, no compression

// Vte-bench, benchmark 1: scrolling. page size: 237 x 55, Monospace Font.

Terminal Emulator	real	user	sys	Notes
Alacritty	0m1,936s	0m0,001s	0m0,946s	
Kitty	0m2,985s	0m0,000s	0m0,567s	
Gnome Terminal	0m3,758s	0m0,001s	0m0,940s	
Terminal ctrl	0m4,064s	0m0,000s	0m0,859s	
Simple terminal (0.8)	0m4,623s	0m0,000s	0m1,105s	
xterm	0m47,249s	0m0,000s	0m0,567s	

// Vte-bench, benchmark 2: alt-screen-random-write, page size: 237 x 55, Monospace Font.

Terminal Emulator	real	user	sys	Notes
Alacritty	0m1,520s	0m0,000s	0m0,554s	
Kitty	0m2,750s	0m0,000s	0m0,408s	
Gnome Terminal	0m3,153s	0m0,004s	0m0,726s	
Terminal ctrl	0m3,790s	0m0,000s	0m0,474s	
Simple terminal (0.8)	0m7,097s	0m0,001s	0m0,825s	
xterm	0m19,797s	0m0,001s	0m0,993s	

// Vte-bench, benchmark 3: scrolling-in-region, page size: 237 x 55, Monospace Font.

Terminal Emulator	real	user	sys	Notes
Alacritty	0m3,337s	0m0,001s	0m3,297s	
Gnome Terminal	0m5,280s	0m0,001s	0m3,254s	
Kitty	0m6,326s	0m0,001s	0m6,034s	
Terminal ctrl	0m6,739s	0m0,001s	0m4,033s	
Simple terminal (0.8)	0m7,844s	0m0,001s	0m7,348s	
xterm	3m36,599s	0m0,001s	0m11,011s	

(Eventually I will add the final benchmark results to the Terminal package as a markdown file. Consider the above mentioned results only as a preview.)

If you have any questions, feature requests, bug reports, suggestions, criticism, etc. let me know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 09 Jun 2020 21:48:27 GMT
View Forum Message <> Reply to Message

Hello,

Terminal ctrl's TURTLE backend support was broken for some time.
The reason: Apparently there is no Upp::Append() method for image drag and drop (Or I am missing something...).
I've committed a patch to workaround this problem by disabling image drag-and-drops in Turtle backend.

A good news is that it is now possible to compile Terminal Ctrl for linux framebuffer, using the LinuxFrameBuffer package.

There are some rough edges (e.g. the mouse wheel does not work ATM), but it is basically working on linux framebuffer too. :)

However, currently the support for linux framebuffer is "unofficial". If you are going to try it at all, TRY IT AT YOUR OWN RISK.

I will focus on these backends after the V0.4 release...

Best regards,
Oblivion.

Subject: Re: A terminal emulator widget for U++
Posted by Zbych on Wed, 10 Jun 2020 22:13:56 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Tue, 09 June 2020 23:48
A good news is that it is now possible to compile Terminal Ctrl for linux framebuffer, using the LinuxFrameBuffer package.
There are some rough edges (e.g. the mouse wheel does not work ATM), but it is basically working on linux framebuffer too. :)

Hi Oblivion,

I've added wheel event to framebuffer package. Please check it.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 11 Jun 2020 09:08:08 GMT
View Forum Message <> Reply to Message

Hello Zbych,

Thank you for your efforts!

Indeed, it is working now. :)

Since you are here, allow me to point out two problems I've encountered while testing both terminal package and others (uword, for example).

- Text mode is not disabled on linuxframebuffer based apps when they are run from the linux (tested with v5.6) console. (The framebuffer display is overwritten by text and messages). The "fix" seems to be setting the STDINPUT to KD_GRAPHICS and resetting it to KD_TEXT before the app exits (using ioctl). But since I don't know the specifics of LinuxFrameBuffer package, I can't give more info as to where the problem lies.

- Both on UWord and Terminal package, the return/enter keys are sending the keycode twice (or so it appears: text cursor is not moved to next line but two lines below.)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Wed, 17 Jun 2020 22:36:24 GMT
View Forum Message <> Reply to Message

Hi,

I'm still working on range-based font substituiton feature. And I have finally prototyped the mechanism.

The screenshot below uses the legacy demo.txt file for demonstration. It displays four (default + additional three) fonts for different glyph ranges on the same page.

Font configuration:

IBM Plex Mono: 0x0020-0x07FF (Base glyphs)
Fira Code Mono: 0x2500-0x259f (Box drawing glyphs)
GNU Free Mono: 0x2800-0x28FF (Braille glyphs)
Awesome   Mono: ---------- (Other glyphs)

It has its rough edges, but "it works" fine with a negligable performance hit. :)

Note that this feature will not be available before v0.4.

Best regards,
Oblivion

File Attachments

1) TerminalCtrl_RangeBasedFonSubstitution.png, downloaded 1094 times

---

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 21 Jun 2020 17:35:39 GMT
View Forum Message <> Reply to Message

Hello,

A screengrab of Terminal ctrl on Linux framebuffer. :)

Note: This is only an experimental feature yet to be refined and still unofficial.

However, Thanks to Zbych, the LinuxFrameBuffer package has been updated, and line feeds are working as expected now.
(There are other similar problems but I'm noting them down, so that I can provide feedback.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Zbych on Sun, 21 Jun 2020 20:16:34 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Thu, 11 June 2020 11:08
- Text mode is not disabled on linuxframebuffer based apps when they are run from the linux

(tested with v5.6) console. (The framebuffer display is overwritten by text and messages). The "fix" seems to be setting the STDINPUT to KD_GRAPHICS and resetting it to KD_TEXT before the app exits (using ioctl)

I did some research and it appears that turning keyboard off in virtual terminal is correct solution. Unfortunately neither ioctl(tty_handle, KDSKBMUTE, 1) nor ioctl(tty_handle, KDSKBMODE, K_OFF) solves the problem with keys leaking to the console.
They simply return error "Operation not permitted".

In case you want to do more tests:
https://chromium.googlesource.com/chromium/chromium/+/trunk/ui/ozone/platform/dri/virtual_terminal_manager.cc
https://github.com/LuaDist/sdl/blob/master/src/input/evdev/S DL_evdev.c#L395

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 21 Jun 2020 20:36:11 GMT
View Forum Message <> Reply to Message

Hello Zbych,

Quote:
I did some research and it appears that turning keyboard off in virtual terminal is correct solution. Unfortunately neither ioctl(tty_handle, KDSKBMUTE, 1) nor ioctl(tty_handle, KDSKBMODE, K_RAW) solves the problem with keys leaking to the console.
They simply return error "Operation not permitted".

Again, thank you very much for your efforts!
Also thanks for the links.

I am going to look into LinuxFrameBuffer code in the following days and if I can come up with a workaround or solution, I'll send a patch to review.

By the way, did you try to set STDIN_FILENO to KD_GRAPHICS too? because IIRC, that (Setting both input and output to graphics mode) was the trick used by an app whose name I forgot now, some time ago.)

I'd really love to see Terminal ctrl run on linux frame buffer, because then I can get rid of unnecessary cruft on my personal servers.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Zbych on Sun, 21 Jun 2020 20:47:03 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sun, 21 June 2020 22:36
By the way, did you try to set STDIN_FILENO to KD_GRAPHICS too?

Do you mean using STDIN_FILENO in ioctl(STDIN_FILENO, KDSETMODE, KD_GRAPHICS)?
it returns error: Inappropriate ioctl for device (25)

In my tests I pass path to virtual terminal manually (for example /dev/tty3), I decided to leave tty number autodetection as next step.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 21 Jun 2020 20:54:33 GMT
View Forum Message <> Reply to Message

Yeah, but I am not sure as to how, ATM. (Maybe I am not remembering it correctly).
I'll do some investigation and let you know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 25 Jun 2020 06:36:46 GMT
View Forum Message <> Reply to Message

Hello Zbych,

I've -hopefully- solved the "key leakage" problem in LinuxFrameBuffer package. :)
It required sone changes in the existing code and some additions.
I will upload the patched code for you to review on this weedkend.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Zbych on Thu, 25 Jun 2020 18:40:12 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Thu, 25 June 2020 08:36
I've -hopefully- solved the "key leakage" problem in LinuxFrameBuffer package. :)

I am glad you've found the solution.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 28 Jun 2020 09:57:39 GMT
View Forum Message <> Reply to Message

Hello Zbych,

I have attached the patched LinuxFrameBuffer package for you to review. Feel free to comment on it.

Package is compiled with the latest stable GCC and CLANG on Linux v5.4 and v5.6 machines. For testing, I used the supplied LinuxFrameBufferExample (UWord) code, Terminal ctrl, and some other U++ examples made to run on FB backend.

Changes/additions:

- Automatic VT allocation mechanism is implemented.
  Since U++ suppports function keys from F1 to F12, I have limited the valid VT numbers within tty1-tty12 range (This can be easily changed if required).

- Accordingly, VT switching is implemented.
  Since we mute the vt keyboard input, this required manual handling of the linux signals, and as a result the vt handle had to be made a global static integeer (atomic).

- To prevent any screen buffer damage, U++ linuxframebuffer backend now does not copy the drawn image onto framebuffer when the
  vt is switched away (i,e., whem it loses focus).

- Accordingly, CTRL + ALT + [F1-f12] key combinations are reserved for VT switching.

Remaining issues:

- SHIFT keys are not working properly. (Can't get any capital letters.)

- No double click.

- No key repeat.

I am not sure about the security implications though. If you find any problem, let me know.

Best regards,
Oblivion

File Attachments

1) LinuxFrameBuffer.zip, downloaded 380 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 05 Jul 2020 15:40:27 GMT
View Forum Message <> Reply to Message

Hello,

Terminal ctrl has seen a small but important update:

- Handling of incomplete/partial UTF-8 bytes is improved.

The parser now buffers the incomplete UTF-8 bytes encountered at the end of an incoming data chunk and flushes them with the next chunk.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 14 Jul 2020 12:44:27 GMT
View Forum Message <> Reply to Message

Hello,

Moving towards the v0.4, Terminal ctrl has gained a new feature:

- Application clipboard manipulation protocol (OSC 52) is implemented.

This optional feature grants clipboard read and write access to applications that supports OSC 52 OSC 52 is supported by many terminals, including xterm, and apps such as tmux.

However, this feature should be handled with care by the client code, as it may pose a security threat.

Terminal ctrl already takes some security measures though:

1) Terminal ctrl gives the client code a granular control over this protocol.

2) In compliance with the principle of least astonishment, the feature is disabled by default.

3) It will work IFF it is enabled && the ctrl instance has focus (to prevent spamming, etc.)

Accordingly, the API and specs docs are updated.

Best regards,

Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 01 Aug 2020 18:48:20 GMT
View Forum Message <> Reply to Message

Hi,

Aside from small updates, there were no news about Terminal ctrl for some time.

Here is a screenshot of the upcoming v0.4 (2020.2) of Terminal ctrl on Windows 10, scheduled to be published in October 2020:

That's Windows powershell (tm) running natively on the basic terminal example, on Windows 10. :)

This is possible because Windows 10 has a brand new pseudoconsole api. (Thus you can also run plain cmd.exe, or other console apps...)

I am implementing this in PtyProcess class, without requiring any API change, so the Terminal reference examples can be run without changing the code.

Hopefully, this will mature with v0.4.

Best regards,
Oblivion

File Attachments

1) Capture.JPG, downloaded 1229 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 21 Aug 2020 11:29:59 GMT
View Forum Message <> Reply to Message

Hi,

Today marks a rather important milestone for Terminal ctrl: Windows 10 pseudoconsole API support.

The support is implemented in PtyProcess class, so nothing new is introduced. Thanks to Mirek, you only need to enable the WIN10 flag in theIDE's main package configuration settings ("GUI WIN10"). So you will need the nightly builds of Upp to run Windows 10 pty with Terminal ctrl.

It is tested with the bundled CLANG and MSVC19 (MSVC19 requires comdlg32.lib via Windows SDK)

Since the support is experimental at the moment, you can expect some glitches (If you find any, let me know).

Git examples are also updated to reflect the changes. You can check them to see what is changed (or rather, not changed.)

For other, under-the-hood improvements and changes, you can check the git commit history.

Note that Terminal package is still in v0.3 phase. v0.4 (or 2020.2) is still two months away, and will bring in more improvements, and hopefully a refined win10 pty support.)

Here is a screenshot of the stock terminal splitter example, running multiple instances of cmd.exe with resizing support. (You can also run powershell instead):

If you have any questions, suggestions, bug reports, etc., let me know.

Cheers!

Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 05 Sep 2020 14:45:39 GMT
View Forum Message <> Reply to Message

Hi,

There is a project called Monotty text-based desktop environment.
This project, which I've been tracking for some time, has set up a public ssh server where you can test both yout terminal emulator and the monontty desktop prototype.

It is cool stuff (with images made of unicode glyphs, 24-bit color transparency,etc.)

You can test the stock ssh terminal example by setting the url to:

vtm@netxs.online:22

After that you get this:

:)

Consider it also as a demo for the SshShell component of Core/SSH package.

Best regards,

Oblivion.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 01 Oct 2020 16:11:58 GMT
View Forum Message <> Reply to Message

I am considering a rename for Terminal. (Package name won't change. Only the actual widget)

Should we rename it to TerminalCtrl with v0.4 (2020.2)?

If so, should we add Terminal as an alias (for backward compat.)?

What dou you think?

Best regards,
Oblivion

Should Terminal be renamed as TerminalCtrl(total votes: 5)

Yes 5/(100%)
No 0/(0%)

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 13 Oct 2020 10:41:14 GMT
View Forum Message <> Reply to Message

Ok then, since there is no objection, the Terminal widget will be officially renamed as TerminalCtrl from version 2020.2 (0.4) on.

In the meantime, I'd like to share some news about my personal, pet project:

A robust and easy-to-use, multitabbed, cross-platform terminal emulator (which will also have support for SSH2 connections) I am currently writing in tribute to U++ :)

As I've mentioned before, its project codename is "toad" (it may change in the future). It is a pre-alpha software now, but once it hits alpha (0.1) I will release it under BSD license on a separate git repo. (most stuff - sans SSH infrastructure - is already up and working.)

Here is a screenshot (on linux, GNOME 3.38):

If you have any questions, ideas, bug reports, etc. let me know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Klugier on Tue, 13 Oct 2020 18:41:28 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

TerminalCtrl look amazing. Good job!

Right now I am waiting for the Upp DE with Your terminal App ;)

Klugier

Subject: Re: A terminal emulator widget for U++
Posted by hongnod on Thu, 03 Dec 2020 14:05:08 GMT
View Forum Message <> Reply to Message

I am watching your "Terminal" project for a duration of time. It looks amazing. I just want to know if the "toad" out will be something like XShell?

Thanks for your wonderful doing.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 03 Dec 2020 15:25:30 GMT
View Forum Message <> Reply to Message

Hello hongnod.

Thank you for your kind words!

Quote:I am watching your "Terminal" project for a duration of time. It looks amazing. I just want to know if the "toad" out will be something like XShell?

Yes, but Toad will be lighter. The prealpha (unreleased) version already has a (experimental sftp browser).

My focus with Toad is to provide

a) A cross-platform terminal emulator in U++ style (=single directory with few to none external dependencies, that can also be compiled to work on web browsers and on Linux framebuffer)
b) An ssh2 terminal with x11 forwarding.
c) An integrated sftp browser and a multithreaded sftp/scp file transfer client.

Steps a and b is basically done (I'm polishing them)
Step c is a TODO (done ATM only experimentally).

It is already delayed but its initial public beta will hopefully be released in the second quarter of 2021

In the meantime I will continue improving the TerminalCtrl

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by hongnod on Fri, 04 Dec 2020 05:58:39 GMT
View Forum Message <> Reply to Message

Eager to taste it!

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 05 Dec 2020 15:21:40 GMT
View Forum Message <> Reply to Message

Hi,

Terminal package is now tagged as 2020.2 (v0.4)
There's a lot of changes under the hood but here are some highlights:

Terminal package, release: 2020.2, overview:

* Terminal class is finally renamed as TerminalCtrl. (Upp::Terminal will be available as an alias, until 2021.1 (v0.5) release tag).
* Fixed to compile with MSC 19.
* TerminalCtrl now officially supports Windows 10's native pseudoconsole api via the unified, basic interface of PtyProcess class.
* iterm2's inline images protocol is implemented and polished.
* Clipboard manipulation protocol (OSC 52) is implemented.
* Middle-drag (in application mouse-tracking mode) is implemented.
* Right-drag (in application mouse-tracking mode) is implemented.
* Auto-hide option for mouse cursor is added.
* Word selection is implemented. (double-click selection)
* Line selection is implemented. (triple-click selection)
* Text selection mechanism is refactored using the new range-based methods.
* Copy-on-select style text selection is added.

* Text cursor (caret) width is now automatically adjusted to cell width (i.e. reacts to single/double width unciode codepoints).
* Space character can now have overline/underline/strikeout attributes.
* Key handling is further improved across supported platforms.
* Scheduled time callbacks are now properly cleared on object destruction. (MT-fortification)
* Performance/throughput is further improved by removing the now redundant calls to page refresh methods.
* Hyperlink tooltips now display the decoded URL.
* RGBA, CMY and CMYK color text specifications (formatting and scanning) is now supported.
* Brightness component of faint colors is now correctly calculated.
* Jsonization and Xmlization support is added.
* More switches are added to the TerminalCtrl.usc (layout file).

* PtyProcess api docs are added.
* TerminalCtrl::InlineImage structure is documented.
- Terminal overview doc is removed.

Enjoy! :)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 08 Dec 2020 23:54:42 GMT
View Forum Message <> Reply to Message

Hi,

Marking the beginning of new development cycle, a crucial feature that was missing up until now is initially implemented: PC_style function keys.
From now on it is possible to switch between PC and VT-style function keys by using relevant methods programatically, or via the default options menu or the CTRL+SHIFT+P accelarator key.
Many apps, such as tmux use PC-style function keys by default.
This also means that in VT-style function keys mode the F13-f20 keys should be mapped to CTRL+(F1-F8) as expected.

For more information on Pc-style function keys, see:
https://invisible-island.net/xterm/ctlseqs/ctlseqs.html#h2-P C-Style-Function-Keys

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 23 Jan 2021 00:25:27 GMT
View Forum Message <> Reply to Message

Hi,

I have added a new terminal example to the upp-components/Examples section: TerminalLayoutExample.

This new example demonstrates:

- The usage of TheIDE's layout editor to set-up the TerminalCtrl's properties and layout.
- Custom menu (bar & context) creation, and manipulation of TerminalCtrl's standard menu
- The usage of xterm's WindowOps (actions) to set up custom display sizes and modes (fullscreen/maximize/minimize, etc).
- A high performance event loop.
- Binary serialization of TerminalCtrl's storable properties and flags.

Screenshot:

If you have any questions or suggestions, let me know.

Best regards,
Oblivion

File Attachments
1) TerminalLayoutExample.png, downloaded 975 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Mon, 01 Feb 2021 20:18:15 GMT
View Forum Message <> Reply to Message

Hi,

This is going to be an announcement:

I have decided to move several packages from upp-components to UppHub. And the Terminal package is one of them.

So in the following two weeks I will move the package and the example code to its own github repository.

In this way,

1) You will be able to download and update TerminalCtrl source code, examples via TheIDE's package manager, into the upp sandbox (no need to setup any assembly, etc.) with a single and simple click.
2) You won't have to download the whole upp-components repository to get TerminalCtrl and its examples.

This, however, does not mean that upp-components is dead. On the contrary, it will be maintained and synchronized with the upphub version as usual.
Besides, upp-components, and especially Terminal package gets a lot of hits, as it is referenced in various places in terminal emulation scene, and terminal emulator developer portals, and redirects curious people to Ultimate++. And I don't want to lose that. :)
What I will do is to add a notice to the installation instructions and packages list in the upp-components readme file stating that these packages are also available via upphub, in TheIDE - nightly.

If you have any questions, objections, suggestions, etc let me know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Wed, 03 Feb 2021 16:12:30 GMT
View Forum Message <> Reply to Message

Hello,

A small update:

- Application mouse tracking ("x10/x11") mode now sends the correct mouse coordinates on older wire formats where the max x/y coords is limited to 255 cells, 8-bit). This also fixes vim's mouse handling.

- It is now possible to set the text cursor style (block, beam,underline) of TerminalCtrl via TheIDE's layout editor.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 12 Feb 2021 16:42:09 GMT
View Forum Message <> Reply to Message

Hi,

Terminal package and TerminalCtrl is moved to its own repo and UppHub. (It will continue to be available on upp-components too.)
This means that you no longer need to download the whole repo to get TerminalCtrl and its reference examples.
If you have U++ nightly builds (as of February 12), you can download them via UppHub package manager in the TheIDE.

If you have any questions let me know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Klugier on Fri, 12 Feb 2021 20:58:46 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

Glad to hear that it is available on UppHub :) I tested the TerminalExample and found one problems. The path inside terminal is not displayed correctly (it has the same color as terminal area). Here is the screenshot:

Maybe it is somehow related to white them. Please check!

Klugier

File Attachments
1) TerminalCtrlBug.png, downloaded 880 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 12 Feb 2021 21:26:13 GMT
View Forum Message <> Reply to Message

Hello Klugier,

Quote:. The path inside terminal is not displayed correctly (it has the same color as terminal area). Here is the screenshot:

Thanks for reporting this!

However, I am unable to reproduce it on my two machines (Arch Linux, GNOME.) Tried with several different normal/dark themes. It seems to be working. (Even "powerline" styles are working! :))

It is possibly theme related.

If you have time, could you please provide the below information:

- The TERM variable
- The SHELL variable
- Is Adjust to dark theme on? (You can check it via context menu)
- Is Light colors mode on? (You can check it via context menu)
- Have you tried it with other terminals? (e.g. xterm, Konsole, etc.) xterm would be a good criterion to compare.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Klugier on Fri, 12 Feb 2021 22:00:51 GMT
View Forum Message <> Reply to Message

Hello,

It is fine with Konsole and VS Code however, the xterm doesn't display colors correctly. The color tune up doesn't help. The values for $TERM, $SHELL and $PS1 are as follow:

xterm
/bin/bash
\[\033[01;32m\][\u@\h\[\033[01;37m\] \W\[\033[01;32m\]]\$\[\033[00m\]

I think the root cause of the problem is that distro do some overrides.

Klugier

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 12 Feb 2021 22:14:06 GMT
View Forum Message <> Reply to Message

Hello Klugier,

Thanks for the info.

That's what I've suspected, and that's why I asked for you the check it against xterm.

The bash config. sets the ink color to white. (SGR 37). That color is meant to be changed by the user, If the background (which can be overridden by the theme) is the same color, then it'll be unreadble. (unless you explicitly set the color white or the paper color to something else on TerminalCtrl.)

I reproduced it with xterm, GNOME terminal, VTE and kitty. :)
We can call it a bash config/theme clash and a false alarm.

Best regards,

Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 23 Feb 2021 21:57:43 GMT
View Forum Message <> Reply to Message

A small tip:

TerminalCtrl can display PowerLine fonts and glyphs
By default, however, if you install and try to use PowerLine fonts, you will notice that some glyphs are missing. This is because U++ handles its fallback fonts through a compile-time list.
In order to utilize powerline fonts, all you need to do is add the following line to the end of fallback fonts lists [sFontReplacements array], in Draw/FontCR.cpp:

```
{ "PowerlineSymbols", 0x00000000, 0x08000008 },
```

This method can be applied to other fonts as well. Just use uppbox/FontCover app to calculate the font range for your desired font.

If you recompile U++, set up powerline fonts support and configure apps, say vim, you should get a display similar to the below screenshot(taken via stock TerminalLayoutExample):

Best regards,
Oblivion

File Attachments
1) Ekran Görüntüsü - 2021-02-24 00-47-47.png , downloaded 1305 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 27 Feb 2021 21:07:28 GMT
View Forum Message <> Reply to Message

Hello,

tldlr:

In the following weeks TerminalCtrl will finally gain winpty support.
This means that it will be able to run as a local terminal ("natively") on Windows XP/Vista/7/8/10, without requiring Cygwin, Msys2 or Win10 pseudoconsole api. :)

This idea was brewing for some time as I was investigating the possibility of embedding native windows console support in PtyProcess class.

As it turned out, it was suprisingly trivial, thanks to winpty.

Pros:

1. TerminalCtrl does not require any specific hack or intervention to run on Windows as a local (=not remote) console, thanks to its design and WinPty.

2. Cygwin or MSYS2 are not required. Only the libwinpty (winpty.lib or winpty.dll) and winpty-agent is required and they compile on native CLANG compiler and Windows environment.

3. WinPty's API seamlessly integrates with PtyProcess class and it api. As a result, the existing terminal examples or code using PtyProcess class do not require any modification. A simple compiler flag (flagWINPTY) will be enough to switch it on.

4. This means it is possible run cmd.exe, bash.exe, powershell.exe, or any Windows-native console application (such as FAR file manager - see screenshot below) directly on TerminalCtrl/PtyProcess.

5. This also means that TerminalCtrl can be set to run in CygWin and Msys2 environments, and it can easily replace Mintty, which is, AFAIK, bound to Cygwin or Msys2 subsystem.

6. winpty has MIT license. You can freely supply the winpty.lib and winpty-agent with your application.

Cons:

- Nothing much at the moment, really. Winpty has some small quirks but they arent specific to our terminal emulator widget.

Below screenshot is from the Far file manager. It is running natively on the stock TerminalExample, on Windows 7.

If you have any questions, bug reports, suggestions criticism, etc. let me know.

Best regards,
Oblivion

File Attachments
1) , downloaded 1314 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Mon, 01 Mar 2021 22:49:42 GMT
View Forum Message <> Reply to Message

Hi,

TerminalCtrl now officially supports winpty on Windows platform.

This effectively eliminates one of the major "show-stoppers" we've had up until now with

but we already have PTY support for that version of windows)

You have two choice here should you select the WINPTY backend:

1) You can compile winpty-agent, winpty.dll and winpty.lib. These files can be easily created, It is not a complicated process.
2) You can use the existing files in your system (for example you can get them from MSYS2 installation or from the GIT Windows client). Just put the agent and dll in the same directory as your terminal app and link your app against winpty.lib (by activating WINPTY flag).

In the process, I made a so-called "course-correction" which unfortunately slightly breaks the existing TerminalCtrl installations: I have removed the PtyProcess code from TerminalCtrl source

tree and made it into a separate pacakge.
Thus you will need to explicitly add PtyProcess package and its heeader file to your apllications if you want to use it.

E.g.
#include <PtyProcess/PtyProcess.h>

I apologize for this break, but hopefully this will allow easier maintenance and better separation of code.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Wed, 03 Mar 2021 19:45:26 GMT
View Forum Message <> Reply to Message

Hi a small update,

I have already managed to compile and statically link winpty library with stock Upp and the bundled clang compiler with very little wrestling.
Not to mention that winpty has MIT license.. This means we can simply provide the library with the PtyProcess package, and preferably making it the default backend, for easy maintenance and development. This and testing shouldn't take more than a week or so.
Later I can wrestle with MSC. :)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 07 Mar 2021 13:42:16 GMT
View Forum Message <> Reply to Message

Hello,

- PtyProcess package now contains a patched and statically linked version of libwinpty.

- winpty is from now on the default pty backend on Windows platform. No WINPTY flag is required

to activate it.

- Since winpty requires an agent executable, I have also added PtyAgent to both TerminalCtrl nest on UppHub and to upp-components/Core.

- After building the PtyAgent, just move the resulting exacutable (PtyAgent.exe) to your apps executables directory.

- Package is tested with CLANG. MSC will follow...

This means that effectively Terminal package -as a local pty- and its examples are now able to run and work on various versions of Windows.

If you have any questions or suggestions, let me know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 01 May 2021 12:17:34 GMT
View Forum Message <> Reply to Message

Hi,

Terminal package is updated. This update is somewhat big, as the 2021.1 release is imminent. Consider the below changes as a summary of 2021.1 milestone (v0.5) changes/highlights:

Changes:

* Cell padding (horizontal/vertical) is implemented.
* Missing tester methods are implemented.
+ Display renderer is refactored and optimized.
+ Parser (VTInstream) is optimized, using chunking in addition to traditional state machine. It is ~82% faster on average.on a reference machine.
+ Sixel renderer is refactored and optimized. The renderer's raw performance is almost doubled. From 14 to 27 MB/s on a reference machine.
+ An out-of-bounds crash with dynamic colors feature is fixed.
+ Dynamic colors feature is improved. It can now set/reset multiple colors in a single command.
+ Display refreshing strategy is improved.
+ Local echo now correctly calls the display refreshing methods.
+ A crash when selecting wrapped text lines is fixed.
+ Parser now allows OSC sequences to have UTF-8 payload on UTF-8 parsing mode.

+ SGR mouse tracking mode gained a pixel-level tracking variant.
+ UTf-8 titles (xterm extension) are now correctly displayed and reported.
+ On Windows Vista, 7, 8, 10 PtyProcess now allows native console applications via WinPty.
+ On windows, PtyProcess now defaults to statically linked WinPty backend. (Win10 api stays as the recommended option on Windows ver. >= 10)
+ On windows native console (Via WinPty), TerminalCtrl now correctly refreshes the page size after a window size change.
+ Deprecated "Terminal" alias is now removed.

Only a few more compliance tests and cosmetic changes remain. After that, I'll mark this as 2021.1

Suggestions, bug reports, etc .are welcome.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 25 May 2021 11:00:53 GMT
View Forum Message <> Reply to Message

Hi,

TerminalCtrl will be marked as v0.5 (2021.1) next week. This release will have a lot of improvements.

TerminalLayoutExample (the GUI building example) will include some more tricks such as unicode input method (demonstrating the way to use popup windows with TerminalCtrl.)

A preview (gif):

:)

This will be available next week.

Best regards,

Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 27 May 2021 18:39:32 GMT
View Forum Message <> Reply to Message

I usually don't add screenshots or gifs from Windows, but here it is:

TerminalLayoutExample, running on Windows 7 (Winpty/cmd.exe), demonstrating the improved unicode codepoint insertion popup and a simple banner, using echo parser. (will be available next week)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 29 May 2021 16:08:50 GMT
View Forum Message <> Reply to Message

Hello,

Terminal package is tagged as 2021.1 and officially released/updated. It is available via Upphub and upp-components repo.
This version brings many improvements (such as Windows 7/8/10, winpty support) and bug fixes. See my previous posts and git logs. for details.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Novo on Tue, 15 Jun 2021 14:41:32 GMT
View Forum Message <> Reply to Message

Compilation problem on Mac while compiling TerminalInWebBrowserExample:
/Users/ssg/worker0/m-upphub-TerminalCtrl/build/TerminalCtrl/Terminal/Keys.cpp:312:89: error: use of undeclared identifier 'K_OPTION_KEY'; did you mean 'K_OPTION'?
	if(findarg(key & ~(K_CTRL|K_ALT|K_SHIFT|K_OPTION), K_CTRL_KEY, K_ALT_KEY, K_SHIFT_KEY, K_OPTION_KEY) >= 0)
	^~~~~~~~~~~~
	K_OPTION
/Users/ssg/dvlp/cpp/code/upp/git/uppsrc/CtrlCore/CtrlCore.h:103:2: note: 'K_OPTION' declared here
	K_OPTION	= 0x4000000,
	^

MacOS can be easily installed in a VM (check for such projects on github).

Subject: Re: A terminal emulator widget for U++
Posted by Klugier on Fri, 26 Nov 2021 22:19:10 GMT
View Forum Message <> Reply to Message

Hello Oblivion,

I just read the article about 10 Cool Command Line Application on OMG Ubuntu, how many of theme are compatible with TerminalCtrl? I think these applications give good testing environment. You could try it by yourself!

Klugier

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 27 Nov 2021 06:51:54 GMT
View Forum Message <> Reply to Message

Hello Klugier,

(I've been changing my job, so I was away for several month. And I am almost back to developing U++ stuff... Nice to see you all. :))

All of those mentioned applications AFAIK are compatiable with TerminalCtrl. We even have a video demonstrating the mapscii on TerminalCtrl's earlier versions. (I've tested them all except the twitter app) If you find any problems. let me know.

Even the highly experimental next-gen terminal stuff (notcurses demos, vtm, etc.) are compatible wihh TerminalCtrl. And the new wchar32 will make things even better on the unicode side. (I am going to test it soon)

ATM, TerminalCtrl passes 425 unit tests, and fails on 86 tests when passed as 'xterm'. This is AFAIK one of the highest scores for non-xterm-based VTEs. (results are taken from esctest terminal unit testing suite).
Where Gnome Terminal can pass only 136 tests when disguised as 'xterm' (Not that Gnome terminal is bad, it just does not implement some legacy stuff for old apps and other features)

Not to mention that even Far Manager (a CLI-based TUI file manager on Windows) works on TerminalCtrl. 8)

A side note: I use vim and emacs, htop, ranger on TerminalCtrl regularly.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 27 Nov 2021 09:47:26 GMT
View Forum Message <> Reply to Message

Klugier wrote on Sat, 27 November 2021 01:19Hello Oblivion,

I just read the article about 10 Cool Command Line Application on OMG Ubuntu, how many of theme are compatible with TerminalCtrl? I think these applications give good testing environment. You could try it by yourself!

Klugier

A screenshot of btop:

:)

Best regards,
Oblivion

File Attachments
1) Ekran Görüntüsü - 2021-11-27 12-41-44.png , downloaded 997 times

---

Subject: Re: A terminal emulator widget for U++
Posted by slashupp on Tue, 21 Dec 2021 06:39:57 GMT
View Forum Message <> Reply to Message

a lazy man's questions:

1. (I did RTFM - so this question goes away)

2. Can the ctrl be used to run as a login shell? (not just as an emulator)
(This is for applications that require use of .profile and not just .bashrc)

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 21 Dec 2021 08:40:34 GMT
View Forum Message <> Reply to Message

Hello slashupp,

Quote:2. Can the ctrl be used to run as a login shell? (not just as an emulator)
(This is for applications that require use of .profile and not just .bashrc)

Yes you can. TerminalCtrl, being a ctrl, simply interprets a string input which may contain terminal escape commands and characters. You feed it with any source of string.
It, however, comes with an optional/default PtyProcess class, which can runs system processes, including login shells. (If it doesn't suit your needs you can simply replace the pty with any other source)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 12 Feb 2022 18:16:23 GMT
View Forum Message <> Reply to Message

Hi,

Terminal package will receive a big update in March 2022.

In the meantime, however, I am happy to announce that you can enjoy the brand new enhanced unicode support of U++ (wchar32) in TerminalCtrl, which means the terminal can now display virtually all of the emojis correctly.
The required changes were minor, and already pushed to the both upp-components and Upphub repos, as TerminalCtrl was already designed to work with up ot 32 bit characters from the beginning.

A screenshot taken with vtm, a modern text user interface and terminal multiplexer (running on TerminalLayoutExample)

You can access the free demo of vtm to test TerminalCtrl, using the following command via the stock TerminaCtrl examples:

#ssh vtm@netxs.online

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by jjacksonRIAB on Fri, 20 May 2022 09:48:48 GMT
View Forum Message <> Reply to Message

Very nice control but it also needs a ton of optimization work. If I run find . * in my home directory it takes 7.4 seconds in Konsole, ~12 seconds in lxterm and xterm and 1 minute 44 seconds in TerminalCtrl.

Subject: Re: A terminal emulator widget for U++
Posted by jjacksonRIAB on Fri, 20 May 2022 11:53:59 GMT
View Forum Message <> Reply to Message

Just to make sure I tried it again with

tr -dc '[:graph:]' < /dev/urandom | tr -d \"\\\'\` | head -c 1G > test.txt

and

time cat test.txt

For Konsole I got:

real 1m4.362s
user 0m0.017s
sys 0m6.407s

and for TerminalCtrl I got

real 18m33.085s
user 0m0.024s
sys 0m10.896s

rxvt appears to be the fastest I've tested at

real 0m17.997s
user 0m0.004s
sys 0m4.674s

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 20 May 2022 21:24:50 GMT
View Forum Message <> Reply to Message

Hi!

Quote:y nice control but it also needs a ton of optimization work. If I run find . * in my home directory it takes 7.4 seconds in Konsole, ~12 seconds in lxterm and xterm and 1 minute 44 seconds in TerminalCtrl.

Thank you very much for the feedback!

My guess is that you've probably tested it on TerminalExample (which isn't meant to test the real-life performance of TerminalCtrl.
If so, could you re-test it on TerminalLayoutExample, which uses a proper high performance (high throughput) loop?

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by jjacksonRIAB on Fri, 20 May 2022 21:33:15 GMT
View Forum Message <> Reply to Message

Oblivion,

Your guess was correct, I tried the layout example as instructed:

real 0m50.865s
user 0m0.015s
sys 0m6.776s

The performance was better than Konsole. Very nice!

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 20 May 2022 22:00:24 GMT
View Forum Message <> Reply to Message

Hello jacksonRIAB,

Just to give you some idea as to where TerminalCtrl stands:

This is somewhat old, but results (ratio) still hold and --in fact lot better--(I will update the results ASAP, after I merge the new optimizations...)

CPU: AMD FX 6100 - Black Edition (Six-core processor)
Memory: 8 GiB, DDR-3 1800 Mhz
Storage: 128 Gib SSD, 1Tib HDD
Graphics: AMD Radeon R7 240, 2GiB
Monitor: LCD, HD (1080p)
OS/Kernel: Arch/Linux 5.9 (x86_64)
Desktop: GNOME 3.36 (X11)
Framework: Ultimate++ 2020.1 (alpha)
Flags: GUI
Compilers: GCC 10.0.1 CLANG 10.0.0
Application: TerminalExample (with a high througput loop).

Disclaimer: The benchmarks below do not represent any competition.
they don't represent any weak spots of the used terminals.
The sole purpose of these benchmarks is to give an idea as to where
the Terminal ctrl's "engine" performance stands ATM.

Note: All benchmarks are repeated 50 times, automatically.
The performance difference between GCC/CLANG builds is negligeable.
The latest stable versions of these terminals are used (as of May-June 2020).

Results of command: "time find /usr", Total files: 415897, page size: 237 x 55, Monospace Font)

Terminal Emulator	real	user	sys	Notes
Alacritty	0m3,776s	0m1,134s	0m2,637s	Scrollback buffer size: 64k
Terminal ctrl	0m4,126s	0m1,202s	0m2,466s	Scrollback buffer size: 64k
Kitty	0m5,608s	0m1,224s	0m2,763s	Scrollback buffer size: 64k
Gnome Terminal	0m6,785s	0m1,243s	0m3,015s	Scrollback buffer size: 64k
xterm	0m24,368s	0m1,661s	0m3,766s	Scrollback buffer size: 64k

Terminal Emulator	real	user	sys	Notes
Simple terminal (0.8)	0m3,791s	0m1,162s	0m2,533s	Has no scrollback buffer.
Terminal ctrl (0.3)	0m3,971s	0m1,120s	0m2,472s	Scrollback disabled.

Terminal Emulator	real	user	sys	Notes

---------- ---------- ---------- ---------- ----------
Terminal ctrl (0.3) 0m4,380s 0m1,479s 0m2,536s localhost, Scrollback buffer size: 2000 lines, no compression, (SshTerminalExample is used)
PuTTY (0.73) 0m40,828s 0m1,354s 0m3,071s localhost, Scrollback buffer size: 2000 lines, no compression

// Vte-bench, benchmark 1: scrolling. page size: 237 x 55, Monospace Font.

Terminal Emulator	real	user	sys	Notes
Alacritty	0m1,936s	0m0,001s	0m0,946s	
Kitty	0m2,985s	0m0,000s	0m0,567s	
Gnome Terminal	0m3,758s	0m0,001s	0m0,940s	
Terminal ctrl	0m4,064s	0m0,000s	0m0,859s	
Simple terminal (0.8)	0m4,623s	0m0,000s	0m1,105s	
xterm	0m47,249s	0m0,000s	0m0,567s	

// Vte-bench, benchmark 2: alt-screen-random-write, page size: 237 x 55, Monospace Font.

Terminal Emulator	real	user	sys	Notes
Alacritty	0m1,520s	0m0,000s	0m0,554snear-real-life,	
Kitty	0m2,750s	0m0,000s	0m0,408s	
Gnome Terminal	0m3,153s	0m0,004s	0m0,726s	
Terminal ctrl	0m3,790s	0m0,000s	0m0,474s	
Simple terminal (0.8)	0m7,097s	0m0,001s	0m0,825s	
xterm	0m19,797s	0m0,001s	0m0,993s	

// Vte-bench, benchmark 3: scrolling-in-region, page size: 237 x 55, Monospace Font.

Terminal Emulator	real	user	sys	Notes
Alacritty	0m3,337s	0m0,001s	0m3,297s	
Gnome Terminal	0m5,280s	0m0,001s	0m3,254s	
Kitty	0m6,326s	0m0,001s	0m6,034s	
Terminal ctrl	0m6,739s	0m0,001s	0m4,033s	
Simple terminal (0.8)	0m7,844s	0m0,001s	0m7,348s	
xterm	3m36,599s	0m0,001s	0m11,011s	

If you need help with TerminalCtrl, or have any questions, please let me know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by jjacksonRIAB on Sat, 21 May 2022 04:23:25 GMT
View Forum Message <> Reply to Message

Thanks for the data, Oblivion. Impressive work!

Subject: Compile problem on WIN10
Posted by peterh on Tue, 13 Sep 2022 13:15:52 GMT
View Forum Message <> Reply to Message

Hi, I tried TerminalLayoutExample and some other Examples.

These show a compilation error on WIN 10. It doesnt matter if I use CLang or MSVC.

```
HANDLE WinPtyCreateProcess(const char *cmdptr, const char *envptr, const char *cd, winpty_t*
hConsole)
{
 Buffer<wchar> cmd, env;
 sCmdToUnicode(cmd, cmdptr);
 sEnvToUnicode(env, envptr);

 auto hSpawnConfig = winpty_spawn_config_new(
  WINPTY_SPAWN_FLAG_AUTO_SHUTDOWN,
  cmd,                    //error here
  nullptr,
  cd ? ~WString(cd) : nullptr,  //error here
  env,                    //error here
  nullptr);
```

```
C:\upp\UppHub\TerminalCtrl\PtyProcess\Win32Pty.cpp (104): error: no matching function for call
to 'winpty_spawn_config_new'
 (): auto hSpawnConfig =  w inpty_spawn_config_new (
C:\upp\UppHub\TerminalCtrl\PtyProcess/lib/include/winpty.h (170): note: candidate function not
viable: no known conversion from 'Buffer<Upp::wchar>' (aka 'Buffer<unsigned long>') to
'LPCWSTR' (aka 'const wchar_t *') for 2nd argument
 ():  w inpty_spawn_config_new(UINT64 spawnFlags,
 (): 1 error generated.
```

It cannot convert from Buffer<wchar> to LPCWSTR.
I tried to fix it, but no sucess.

My C++ and Upp knowledge is very limited, C knowledge is pretty good.
It happens with Build 16324 and with the stable Build.
Edit:
I can compile it by this modification:

```
auto hSpawnConfig = winpty_spawn_config_new(
 WINPTY_SPAWN_FLAG_AUTO_SHUTDOWN,
 (wchar_t *) ~cmd,
 nullptr,
 (wchar_t *) (cd ? ~WString(cd) : nullptr),
 (wchar_t *) ~env,
 nullptr);
```

Then it compiles and runs, it opens a window, and closes it immediately without any error message.
I dont think this is the intended behaviour.

Subject: Re: Compile problem on WIN10
Posted by Oblivion on Tue, 13 Sep 2022 19:39:56 GMT
View Forum Message <> Reply to Message

Hello peterh,

Thank you for reporting this problem. This is a known problem due to the U++'s transition to 16 bit wchar to 32 bit wchar. There is already a fix in the pipeline and the change will be pushed to the repo this weekend.
It requires a slightly more modification than just type casting (The culprit is env variables structure...)

Best regards,
Oblivion

Subject: Re: Compile problem on WIN10
Posted by peterh on Wed, 14 Sep 2022 01:48:38 GMT
View Forum Message <> Reply to Message

Thank you.
My typecasting was a quickn dirty attempt and only for experimentation to get something running in the debugger for further investigation.
I cant fix it myself, that is clear now.
Thanks for your work!

Subject: Re: Compile problem on WIN10
Posted by Oblivion on Wed, 14 Sep 2022 22:31:57 GMT
View Forum Message <> Reply to Message

Hello peterh,

I've fixed the winpty backend wchar issue (hopefully) and pushed the changes. Please check.

By the way, if you are on Win10 you can also use the native pseudoconsole api of Win10, if it is installed (recommended).
To do that you need to compile U++ with WIN10 flag (flagWIN10)

Best regards,
Oblivion

Subject: Re: Compile problem on WIN10
Posted by peterh on Thu, 15 Sep 2022 04:54:14 GMT
View Forum Message <> Reply to Message

Thank you.

I tried two examples: TabbedTerminalExample and TerminalLayoutExample.
Both show similar behavior:
With "GUI" they compile without warnings. When started, they open a window and immediately close it and terminate without error.

With "GUI WIN10" there is the same compilation error in both:

```
C:/upp/out/UppHub/PtyProcess/CLANGx64.Debug.Debug_Full.Gui.Win10\PtyProcess$blitz.cpp
(13): In file included from
C:/upp/out/UppHub/PtyProcess/CLANGx64.Debug.Debug_Full.Gui.Win10\PtyProcess$blitz.cpp:1
3:
C:\upp\UppHub\TerminalCtrl\PtyProcess\Win32Pty.cpp (80): error: expected ')'
C:\upp\UppHub\TerminalCtrl\PtyProcess\Win32Pty.cpp (88): error: extraneous ')' before ';'
C:\upp\UppHub\TerminalCtrl\PtyProcess\Win32Pty.cpp (81): warning: left operand of comma
operator has no effect [-Wunused-value]
C:\upp\UppHub\TerminalCtrl\PtyProcess\Win32Pty.cpp (82): warning: left operand of comma
operator has no effect [-Wunused-value]
 ():  n ullptr ,
C:\upp\UppHub\TerminalCtrl\PtyProcess\Win32Pty.cpp (84): warning: left operand of comma
operator has no effect [-Wunused-value]
C:\upp\UppHub\TerminalCtrl\PtyProcess\Win32Pty.cpp (87): warning: left operand of comma
operator has no effect [-Wunused-value]
C:\upp\UppHub\TerminalCtrl\PtyProcess\Win32Pty.cpp (88): warning: expression result unused
[-Wunused-value]
```

On Win10 64 Bit, compiled with Clang 64 bit and MSVC 64 bit.
Upp is version 16324. I tried Stable built of Upp too, this did not solve it.

Subject: Re: Compile problem on WIN10

Posted by Oblivion on Thu, 15 Sep 2022 05:14:36 GMT
View Forum Message <> Reply to Message

Quote:I tried two examples: TabbedTerminalExample and TerminalLayoutExample.
Both show similar behavior:
With "GUI" they compile without warnings. When started, they open a window and immediately close it and terminate without error.

Hmm..

For winpty backend:

Have you also compiled the PtyAgent package (It is in the UppHub/TerminalCtrl folder)?
You also need to compile the PtyAgent package and move the resulting PtyAgent.exe into the same folder of your main executable. It is an agent (daemon) that winpty backend talks to.

For Win10:

I will look into the problem tonight, and try to fix the issue.

Also, you can enable the Log in the relevant ptyprocess file by uncommenting the LLOG

Thank you for your patience.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 15 Sep 2022 05:25:12 GMT
View Forum Message <> Reply to Message

OK,
I've fixed the typo with WIN10 backend. (but don't have time to test now. If you encounter any problems let me know, I'll fix it ASAP)

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by peterh on Thu, 15 Sep 2022 05:58:40 GMT
View Forum Message <> Reply to Message

Yes, now both compile without warnings, both with "GUI" and "GUI WIN10".

To achieve this, it was not enough to reinstall, I deleted the "TerminalCtrl" directory and installed fresh.

Both now open a window and close it immediately.
I am not in a hurry; I am just trying this stuff and do not know how this will work, I am just trying.

I am not sure if I am doing something wrong or if I need something additional.
I had the idea to embed a terminal control into my own app, but as said, I am not in a hurry and if I can help or test something I am happy to do so.

Subject: Re: A terminal emulator widget for U++
Posted by peterh on Thu, 15 Sep 2022 07:03:31 GMT
View Forum Message <> Reply to Message

I think I found a problem in TabbedTerminalExample main() Run()

```
  while(IsOpen() && !tabs.IsEmpty()) {
   ProcessEvents();
   for(int i = 0; i < tabs.GetCount(); i++) {
    TerminalTab& tt = tabs[i];
/* Error here? --->*/  if(!tt.Do()) {
     tabbar.RemoveCtrl(tt);
     tabs.Remove(i);
     i--;
    }
   }
   Sleep(10);
  }
```

If I replace the marked line by
if(i && !tt.Do()) {
Then it opens a tabbed window.
I dont fully understand the code and why the loop counter is decremented inside the loop, however this seems to be a problem, because "i" becomes negative, when it was zero before.

Edit: I do now think after some research, the program is ok, and I simply dont have the interface required.
I know terminals and emulators from my job (I am retired now), these start up, then you can select the interface, but this program apparently has no selection and I do not know which interface or port it needs.

Subject: Re: A terminal emulator widget for U++

Hello peterh,

peterh wrote on Thu, 15 September 2022 08:58Yes, now both compile without warnings, both with "GUI" and "GUI WIN10".

To achieve this, it was not enough to reinstall, I deleted the "TerminalCtrl" directory and installed fresh.

Both now open a window and close it immediately.
I am not in a hurry; I am just trying this stuff and do not know how this will work, I am just trying.

I am not sure if I am doing something wrong or if I need something additional.
I had the idea to embed a terminal control into my own app, but as said, I am not in a hurry and if I can help or test something I am happy to do so.

Thank you for your reports and patience.

For winpty backend:

I assume that you've moved the PtyAgent.exe into the same directory as TerminalLayoutExample.exe,

Then "normally" there are only a few things that can cause the terminal window to open and close immediately (pty failure):

1) The variables passed to PtyProcess (env, cd, cmd) have some error. E.g., the examples use "ComSpec" env variable to locate the command line interface executable (cmd.exe or powershell.exe, etc.) This can be checked by manually providing the full path of cmd.exe or powershell.exe to the PtyProcess.

2) Execution of or access to PtyAgent executable is blocked by Windows. (insufficient permissions)

Either case, I'd be grateful if you could post the log file (uncommenting the LLOG() macro in Win32Pty.cpp before running the example)

Still, I am going to improve the error handling/diagnostics in PtyProcess this weekend.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by peterh on Fri, 16 Sep 2022 04:10:31 GMT
View Forum Message <> Reply to Message

Quote:I assume that you've moved the PtyAgent.exe into the same directory as TerminalLayoutExample.exe,

I did not know about PtyAgent.exe and searched my whole upp folder for it, no such file exists.
I searched also for PtyAgent and discovered a package with this name.
I tried to compile it, but got errors:
I do yet not know the reason, I must dig deeper in this, in the moment I do not know enough to ask qualified questions about it....

Subject: Re: A terminal emulator widget for U++
Posted by peterh on Fri, 16 Sep 2022 05:41:02 GMT
View Forum Message <> Reply to Message

Ok, now I compiled PtyAgent.exe and it runs in mode "GUI", but not in mode "GUI WIN10".
In this later case the logfile is:

* C:\upp\out\UppHub\CLANGx64.Debug.Debug_Full.Gui.Win10\TerminalLayoutExample.exe 16.09.2022 07:37:46, user: peter

PtyProcess [WIN32]: Win32CreateProcess() failed.
PtyProcess [WIN32]: Couldn't set pty size!

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 17 Sep 2022 16:40:22 GMT
View Forum Message <> Reply to Message

Hello peterh,

I've tried to fix WIN10 issue. Please check.

The lack of documentation is my fault (laziness/). I will add some doc to clarify the usage and setup of PtyProcess.
In the meantime, here's a short summary that can make basic things clear.

1) PtyProcess can use two backends on Windows (for console process):

a) winpty: This backend can be compiled and run on Win >= Vista.
Requires the PtyAgent.exe to be present. This agent is a daemon that handles the win32 console stuff.
The PtyAgent executable must be put into the same directory as the application using the

PtyProces.
PtyAgent can be found in TerminalCtrl/PtyAgent folder. It has to be separately compiled.
This backend does not require U++ to be compiled with WIN10 flag.

b) win10: This backend uses the window 10 native pseudoconsole api. This api and relevant subsystem is added to
windows with the later revisions of windows 10. Does not use or require winpty or Ptyagent.exe
This backend requires U++ to be compiled with WIN10 flag.

2) Either of these backends can be used on windows version >= 10. Only winpty can be used on Windows version < 10

3) TerminalCtrl does not require PtyProcess. You can write your own. TerminalCtrl only expects data input.
The source of the input is up to the user/developer. PtyPRocess is only a reference implementation,
and provided with the Terminal packages as a default option.

4) To better test TerminalCtrl and PtyProcess in Windows, recommended demo case would be to install
the good old Far Manager (a text mode file manager you can find on the net).

If you have more questions or need any help embedding TerminalCtrl/PtyProcess into your app, let me know.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by peterh on Sat, 17 Sep 2022 18:58:09 GMT
View Forum Message <> Reply to Message

Thank you very much.
It is now functional in both modes "GUI" and "GUI WIN10".
Also your description made it clear enough to find my way.

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 20 Nov 2022 20:11:26 GMT

Hi,
Next week I will push the latest changes to Terminal package.
On of the major highlights of this version will be the performance optimizations.
I have heavily refactored the sequence parser and have been testing the hell out of it for around 6 months.
The results are: 3-5 times higher throughput.(In certain demanding scenarios the jump is from 120 MB/secs, to 600 MB secs, which is almost the limits of the testing machine (specs will be available with the update too)

But for now I have used the very demanding notcurses demos to whet your appetite (A video):

Notice that, the application (notcurses demo) in not simply bombarding the terminal display with a bunch of unicode characters to sample a video. As you can see on the top-right corner, there is actually a sixel "camera" view of the same lift-off video, in sync. The raw "horse-power" of the optimized parser and sixel renderer is enough for us to both write unicode (notascci) art image and and direct pixels on the terminal display at the same time. And this happends synchronously, no MT is used.

Link to the full video (taken on TerminalLayoutExample): https://vimeo.com/773086733

Best regards,
Oblviion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 31 Dec 2022 10:19:13 GMT

Hi,

Parser and sixel optimization has landed in TerminalCtrl.

And the age old question finally has an answer: But, can it run DOOM?

Yes, yes, it can!..

8)

More on: https://vimeo.com/manage/videos/785369882

Happy new year to all U++ users.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 20 May 2023 15:15:58 GMT
View Forum Message <> Reply to Message

Hi,

Terminal package has seen some improvements and fixes.

- Terminal lines are no longer truncated on window/terminal resize (when the new size is smaller than the old size).
- Size hint now has round edges.
- Programmable colors (reset) wrong index variable fixed.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 17 Dec 2023 14:04:44 GMT
View Forum Message <> Reply to Message

Hi, a minor addition to TerminalCtrl:

It is now possible to override the application mouse tracking mode by using a configurable modifier key (ctrl, shift, alt and their combinations).

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 21 Dec 2023 23:23:52 GMT
View Forum Message <> Reply to Message

Hi,

TerminalCtrl has gained another important feature:

Custom highlighting support: Client code can now manipulate the visible terminal lines to add custom highlighting to TerminalCtrl.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 06 Jan 2024 20:45:31 GMT
View Forum Message <> Reply to Message

Hi,

Another weekly update.
TerminalCtrl has gained text search support. Client code can now add any type of custom text search method to TerminalCtrl.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 13 Apr 2024 08:42:27 GMT
View Forum Message <> Reply to Message

Hi,

After a long, silent period, we're back here again.

There has been much work going into TerminalCtrl nowadays, mostly bug fixes, optimizatios and internal maintenance but also some cool new features are here too. Since the next "release" (milestone) will be 2024.1 (v0.9), I'd like to give an update on its status.

Bug fixes aside, there are now six important features, already fully or partially implemented:

1) Selector Mode: TerminalCtrl now allows full keyboard navigation for selecting text in a special mode called "Selector Mode".
 At the moment, three types of selection are possible in this mode: text, rectangle, word.

2) Annotations: This is a cool new feature. User or the hosted app can now add annotations to terminal texts (in both plain text and rich text (qtf) format).
 This can be done via user input (mouse, or keyboard in selector mode) or programatically (using a simple, proprietary protocol). AFAIK, this has been one of the most favored iterm2 features.

3) Word Selection filters: Word selection filter can now be set by the client code. It allows cell-level inspection, so it is even possible to select blocks of text ("words") according to their SGR attributes, associated data type, or colors.

4) Highlighting: TerminalCtrl allows full cell-level highlighting.

5) Shell integration: TerminalCtrl now allows hosted apps or shells to notify the terminal of the working directory, so the terminal can be aware of its environment (and act accordingly).

6) Configurable Keyboard shortcuts: Keyboard shortcuts are now configurable and can be redefined in client code.

All of these features are already utilized in Bobcat, cross-platform terminal emulator.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 11 May 2024 17:00:58 GMT
View Forum Message <> Reply to Message

Hi,

TerminalCtrl has gained a new functionality that is used by many modern terminal emulators: Progress notification and tracking protocol:

OSC 9 ; 4 ; [state] ; [value] ST

This protocol is also supported by Windows Terminal, ConEMU, mintty

Best regards,

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Thu, 15 Aug 2024 19:31:43 GMT
View Forum Message <> Reply to Message

Here is a short video demonstrating the basic capabilities of Bobcat, the terminal emulator (based on TerminalCtrl):

https://vimeo.com/999236026

What is demonstrated in the video, you may ask?

1 - Running Doom in terminal emulator
2 - Navigating the open terminals, using the Navigator.
3 - Using the Finder to find text.
4 - Navigating through the open terminals using l/r keys.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by koldo on Fri, 16 Aug 2024 05:38:55 GMT
View Forum Message <> Reply to Message

It's insane what you can do with your terminal. 8)

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Mon, 30 Sep 2024 21:35:15 GMT
View Forum Message <> Reply to Message

Quote:It's insane what you can do with your terminal. Cool

Ah, I haven't seen your reply, sorry.

Well, thanks to U++, yes. It has become a reasonably powerful terminal emulator. :d 8)

Now it also officially supports Turtle for remote connections and SDL2-GL for headless gui mode (though, it is experimental for now).

A screenshot on SDLGUI backend:

Rumour has that I am also secretly building an SSH terminal, with virtually the same code-base ("Tomcat") ;)

Best regards,
Oblivion

File Attachments
1) bobcat-sdl2gl.png, downloaded 463 times

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sat, 12 Oct 2024 22:18:07 GMT
View Forum Message <> Reply to Message

Hi,

There is another significant improvement in TerminalCtrl package, specifially in PtyProcesss package: PtyProcess package refactored, and process types are split into their respective classes.

- All PtyProcess classes are now derived from APtyProcess base class.
- On Windows, it is now possible to use both WinPty (as WinPtyProcess) and Windows console api (ConPtyProcess) at the same time
- On Linux the actual class is named as PosixPtyProcess.
- Existing code will work, since the PtyProcess is now an alias which designates PosixPtyProcess on *NIX systems, and depending on the WIN10 flag it designates either ConPty or WinPty. (Old behavior is kept.)
- Since they are all derived from the APtyProcess class, it is now possible to dynamically create and store them in the same container.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 08 Dec 2024 09:23:45 GMT
View Forum Message <> Reply to Message

[code]
Hi,

The day has arrived, and TerminalCtrl 2025.1 is released. :)

Release Notes:

Requirements
- U++ >= 2024.1
- C++17

Customizability
- Custom Word/Cell Selection: Supports client-defined word selection via a filter. Selections can be based on text, SGR attributes, data types, or colors.
- Selector Mode: Introduced "Selector Mode" for keyboard navigation and text selection, supporting text, rectangle, line, and word selections.
- Configurable keyboard Shortcuts: All shortcuts are made configurable and can be redefined in client code.
- Text Search: TerminalCtrl now have support for text search, including range-based and parallel (`CoFind`) search
- Annotations: Users and hosted apps can now add annotations to terminal text (plain text or rich text via a simple protocol or menu).
- Custom Highlighting: Custom *cell-level* highlighting support.
- Application Mouse Tracking Override: Added the ability to override application mouse tracking with customizable modifier keys (Ctrl, Shift, Alt, and their combinations).

Protocols
- Directory Change Protocol: Implements the `OSC 7` and `OSC 9 ; 9`protocols to notify the terminal of the shell's working directory.
- Message Notification Protocol: Implements the `OSC 9 ; 2` message notification support.
- Progress Notification Protocol: Implements the `OSC 9 ; 4` protocol, used by modern terminal emulators, this protocol allows tracking of progress from hosted applications.
- iTerm2 Background Image Protocol: Implements iTerm2's background image change protocol.

Appearance
- Hyperlink Underline: Hyperlinks now switch to an under-dot pattern when using the underline attribute.
- Size Hint: The size hint now has rounded edges for better aesthetics.

Bug Fixes
- Mouse Reporting: Fixed mouse reporting bug.
- PC-Style Keys: Fixed the handling of HOME/END keys in PC-style mode.
- Stray Unicode Character: Fixed the Alt+Win key combination issue that produced stray Unicode characters.
- Background Transparency: Addressed transparency and background rendering issues.
- Selection on Resize: Selections are now cleared properly when resizing the terminal.

Performance Improvements
- Renderer Optimization: Refactored the renderer to combine non-contiguous characters, resulting in better performance

---

Behavior Changes
- Modifier Keys: `AltGr` and `Super` modifier keys no longer scroll the page.
- Page Scroll on Paste: The page now automatically scrolls to the bottom when pasting.
- Terminal Resize Behavior: Terminal lines are no longer truncated when resizing to a smaller window.

Code Maintenance
- Code Cleanup: Removed unused variables, redundant code.
- APtyProcess Base Class:
  - All PtyProcess classes are now derived from APtyProcess.
  - On Linux, the active class is `PosixPtyProcess`.
  - On Windows, it can use either `WinPtyProcess` or `ConPtyProcess` based on system configuration.
  - This makes it possible to dynamically manage different PtyProcess types in a single container.

If you wonder what can be achieved with TerminalCtrl, just check Bobcat (which is to be released officially before the end of the year.).

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Lance on Sat, 14 Dec 2024 16:12:57 GMT
View Forum Message <> Reply to Message

great job, Oblivion!

Subject: Re: A terminal emulator widget for U++
Posted by koldo on Sat, 14 Dec 2024 19:50:23 GMT
View Forum Message <> Reply to Message

Oh, it's right

I love this development and how our partner maintains, improves and publishes it.
Long live U++ applications! :)

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Tue, 01 Apr 2025 07:29:52 GMT
View Forum Message <> Reply to Message

Hi,

New feature is added to Terminal/Ptyprocess:

PtyWaitEvent class

- Its API is virtually identical to SocketWaitEvent but for Pty processes.
- Provides a mechanism for waiting on multiple pseudoterminal (pty) processes to become ready for I/O operations.
- On windows it uses I/O Completion Ports (IOCP) to wait for events on process and pipe handles.
- On POSIX-compliant operating systems it uses the poll() system call to wait for events on pty/file descriptors
- On all platform this class effectively removes the limitation of max 64 wait objects.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 13 Apr 2025 12:08:19 GMT
View Forum Message <> Reply to Message

Hi,

TerminalCtrl has gained another capability, another feature: Semantic Information Protocol, or OSC 133

Sequence	Description	Device Level
`OSC 133 ; [command] ST`	Sets the semantic information, starting from the cursor position.	Level 1

Notes

- `command` can be one of the following four values (case sensitive):
  - `A`: Marks the starting point of the shell prompt.
  - `B`: Marks the end of the shell prompt and the start of the user input.
  - `C`: Marks the end of the user input and the start of the command output.
  - `D`: Marks the end of the command output.

- TerminalCtrl currently supports only a minimal--but reasonable--subset of this protocol. This may change in the future.
- TerminalCtrl does not process or display semantic information by itself. Instead, it is up to the client code to make use of the protocol, typically in combination with features like cell highlighting or search functionality.

Well, what is it good for, you might ask.

It allows clear separation between three semantically different sections of terminal output, usually on shells: prompt, user input and command output.
By this separation, it becomes possible to treat three different section of terminal output separately. For example, a search function can search exclusively for command outputs, user inputs or prompts.
Or extracting and highlighting only a specific type of data (e.g. command outputs) become possible.

Of course, Bobcat will utilize this protocol ASAP.

Best regards,
Oblivion

Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Sun, 17 Aug 2025 13:48:55 GMT
View Forum Message <> Reply to Message

Hi,

Happy to announce that TerminalCtrl 2025.2 (v1.0) is released.

What's Changed

Requirements
- U++ 2025.1
- C++17

Highlights

- `PtyWaitEvent` class added for proper event waiting across the supported platforms.
- `PtyProcess` class now check for the dll version of ConPty first, to take advantage of the newest features and improvements.
- A subset of semantic information protocol (`OSC 133`) is implemented.
- Option to treat ambiguous width characters as wide characters.

Changes and Fixes

TerminalCtrl: Semantic information protocol (OSC 133) (59c332c)
TerminalCtrl: Synchronized output feature detection. (7013fa8)
TerminalCtrl: VT mouse events are now ignored if the terminal is in read-only mode. (37bb880)
TerminalCtrl: Clipboard access protocol (write) is advertised in DA. (4f62f0f)

TerminalCtrl: XTVERSION sequence fix (thanks @j4james) (1f3ff94)
TerminalCtrl: Extendended Color format parsing refactored and optimized. (17c737c)
TerminalCtrl: DECRPM 2027 added (reports as permanently reset). (2b1a863)
TerminalCtrl: Parser optimization (d051a38)
TerminalCtrl: VTCell: IsSpecial() method added. VTPage: Minor optimizations. (5af5a33)
TerminalCtrl: Unused insert unicode menu item removed (should be implemented by client code) (6b5d9b1)
TerminalCtrl: Minor renderer optimizations. (bdbd35f)
TerminalCtrl: SetEmulationLevel & SetDeviceConformanceLevel methods are refactored. (511eb34)
TerminalCtrl: RefreshDisplay() method refactored and optimized. (a5b88ee)
TerminalCtrl: Background paint (inverse video) fix. (7b8deb8)
TerminalCtrl: Line length calculation is corrected. (c27c437)
TerminalCtrl: VTPage: Line offset calculation is corrected. (67fa91a)
TerminalCtrl: Option to treat ambiguous chars as wide (2 column) chars. (0f856ee)
TerminalCtrl: Width table updated (ambiguous width chars added). (abc7e6b)
TerminalCtrl: Width table updated (formatting chars table added). (4e6fb72)
TerminalCtrl: Device ID can be changed by the client. (09dff1d)
TerminalCtrl: Report extendend device attributes (term id). (185c9ac)
TerminalCtrl: Selector Mode now automatically initializes to text mode if the mode is unspecified (proper init). (6ecda9d)
TerminalCtrl: Examples are updated to reflect PtyWaitEvent integration & readme updated. (0a2ce7a)

PtyProcess: Windows ConPty now loads as dll. (1ccd90b)
PtyProcess: PtyWaitEvent class is added. Read method (win) refactored. (2fb3e25)

TerminalCtrl is the powerful VT engine behind Bobcat terminal emulator.

Enjoy!