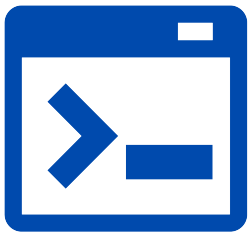


CPP-TERMINAL

CROSS-PLATFORM TERMINAL LIBRARY

1.0.0

Generated by Doxygen 1.10.0

Tue Apr 9 2024 02:24:30

1 cpp-terminal

`CPP-Terminal` is a small and dependency-free C++ library for writing platform independent terminal-based applications. It follows the "Zero-overhead principle" and limits externally included files to the C++ STL. Being cross-platform we are currently supporting Windows, Linux and MacOS and are providing an unified API across all platforms. Our main features are consisting of Colors, Keyboard input, terminal resize handling, as well as other common terminal functionality. It's also possible to open a managed terminal from a windows GUI application.

1.0.1 Hello World example

To write a simple Hello World program, all you need to do is:

```
#include "cpp-terminal/terminal.hpp"
```

```
#include <iostream>

int main()
{
  std::cout « "Just including terminal.hpp activate \033[31mcolor\033[0m !" « std::endl;
}
```

or

```
#include "cpp-terminal/terminal.hpp"
#include "cpp-terminal/color.hpp"
#include <iostream>

int main()
{
  std::cout « Term::color_fg(Term::Color::Name::Red)«"Hello world !"«color_fg(Term::Color::Name::Default)«
      std::endl;
}
```

On windows you can simply create or attach a console through a GUI application by doing:

```
#include "cpp-terminal/terminal.hpp"
#include "cpp-terminal/color.hpp"
#include <iostream>
#include <windows.h>

int __stdcall WinMain(HINSTANCE hinst, HINSTANCE hprev, LPSTR cmdline, int show)
{
  std::cout « Term::color_fg(Term::Color::Name::Red)«"Hello world !"«color_fg(Term::Color::Name::Default)«
      std::endl;
  return 0;
}
```

Until 2021, CPP-Terminal used to be a single header library. Now, CPP-Terminal consists out of multiple small and usage oriented headers:

- `cpp-terminal/input.hpp`: functions for gathering input
- `cpp-terminal/prompt.hpp`: some variations of different prompts
- `cpp-terminal/window.hpp`: a fully managed terminal window for terminal user interfaces (TUI)
- `cpp-terminal/version.hpp`: macros with cpp-terminal's version number

CPP-Terminal tries to be a small and simple replacement for ncurses. This approach keeps the code small and maintainable, but also easy to extend it's functionality. We limit ourselves to a subset of features that work on all supported platforms without needing to worry about style differences or other changes. Any application written with `CPP-Terminal` will work everywhere out of the box natively, without emulation or extra work. The small codebase makes CPP-Terminal easy to debug and extend, as well as understanding what happens behind the scenes in the library's core.

1.0.2 Examples

We have created serval examples to show possible use cases of CPP-Terminal and to get you started more quickly. Every example works natively on all platforms in the exact same way:

- `colors.cpp`: basic color, style and unicode demo
- `kilo.cpp`: the `kilo` text editor ported to C++ and `CPP-Terminal` instead of using Linux specific API
- `menu.cpp`: An interactive menu using only the contents of `cpp-terminal/base.hpp`
- `menu_window.cpp`: An interactive menu using the fully managed windowing system from `cpp-terminal/window.hpp`
- `keys.cpp`: Interactively shows the keys pressed

1.0.3 Supported platforms

Platform	Supported versions	Arch	Compiler	C++ standard
Windows	10 and higher*	x86, x86_64	MSVC 2019, MSVC 2022, clang11, clang12, clang13, clang14, clang15, clang-cl	11,14,17,20
(Windows) MSYS2	All supported	x86, x86_64	ucrt , clang, mingw	11,14,17,20
MacOS	11		xcode11.7 xcode12.4 xcode12.5.1 xcode13 gcc10 gcc11 gcc12	11,14,17,20
MacOS	12		xcode13.1 xcode13.2 xcode13.3 xcode13.4	11,14,17,20
Linux	All supported	x86_64	4.7<=GCC<= 12 3.5<=Clang<=15 intel-oneapi	11,14,17,20
Linux (dockcross)	All supported	arm64 armv5 armv5-musl armv5-uclibc armv6 armv7a, mips, mipsel-lts, s390x, ppc64le, xtensa-uclibc, x86, x64, x64-clang, x64-tinycc	4.7<=GCC<= 12 3.5<=Clang<=15	11,14,17,20

Windows versions prior Windows 10 are missing the Win32 API functionality for entering the "raw mode" and therefore won't work. They are also lacking ANSI support. See #173 for adding support to prior windows versions for MSVC / Win32.

1.0.4 How to use

Adding CPP-Terminal to your own project is really easy. We have collected various ways with easy how-to's in our documentation.

1.0.5 Documentation

Online PDF

1.0.6 Contributing

Contributing to CPP-Terminal is highly appreciated and can be done in more ways than code. Extending it's functionality, reporting or fixing bugs and extending the documentations are just a few of them.

1.0.7 License

1.0.8 Projects using cpp-terminal

- Lime

1.0.9 Similar Projects

Colors

Libraries to handle color output.

C++:

- rang

Drawing

JavaScript:

- node-drawille

Prompt

Libraries to handle a prompt in terminals.

C and C++:

- readline
- libedit
- linenoise
- replxx

Python:

- python-prompt-toolkit

General TUI libraries

C and C++:

- curses and ncurses
- Newt
- termbox
- FTXUI
- ImTui

Python:

- urwid
- python-prompt-toolkit
- npyscreen
- curtsies

Go:

- gocui
- clui
- tview
- termbox-go
- termui
- tcell

Rust:

- tui-rs

JavaScript:

- blessed and blessed-contrib

2 Private folder for cpp-terminal

3 Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

4 Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

5 Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

6 File Index

6.1 File List

Here is a list of all files with brief descriptions:

7 Namespace Documentation

7.1 Term Namespace Reference

Namespaces

- namespace Private
- namespace Version

Classes

- class Argc
- class Arguments
- class Buffer
- class Button
- class Color
- class Cursor
- class Event
- class Exception
- class Focus

  Class to return the focus of the terminal.
- class IOStreamInitializer
- class Key
- class MetaKey
- class Model
- class Mouse
- class Options
- class Screen
- class Terminal
- class TerminalInitializer
- class Terminfo
- class TIstream
- class TOstream
- class Window

  Represents a rectangular window, as a 2D array of characters and their attributes.

Enumerations

- enum class Option : std::int16_t {
  Raw = 1 , Cooked = -1 , ClearScreen = 2 , NoClearScreen = -2 ,
  SignalKeys = 3 , NoSignalKeys = -3 , Cursor = 4 , NoCursor = -4 }

  Option to set-up the terminal.
- enum class Result {
  Yes , No , Error , None ,
  Abort , Invalid }
- enum class Result_simple { Yes , No , Abort }

- enum class Style : std::uint8_t {
  Reset = 0 , Bold = 1 , Dim = 2 , Italic = 3 ,
  Underline = 4 , Blink = 5 , BlinkRapid = 6 , Reversed = 7 ,
  Conceal = 8 , Crossed = 9 , Font0 = 10 , ResetFont = 10 ,
  Font1 = 11 , Font2 = 12 , Font3 = 13 , Font4 = 14 ,
  Font5 = 15 , Font6 = 16 , Font7 = 17 , Font8 = 18 ,
  Font9 = 19 , Font10 = 20 , DoublyUnderlinedOrNotBold = 21 , ResetBold = 22 ,
  ResetDim = 22 , ResetItalic = 23 , ResetUnderline = 24 , ResetBlink = 25 ,
  ResetBlinkRapid = 25 , ResetReversed = 27 , ResetConceal = 28 , ResetCrossed = 29 ,
  DefaultForegroundColor = 39 , DefaultBackgroundColor = 49 , Frame = 51 , Encircle = 52 ,
  Overline = 53 , ResetFrame = 54 , ResetEncircle = 54 , ResetOverline = 55 ,
  DefaultUnderlineColor = 59 , BarRight = 60 , DoubleBarRight = 61 , BarLeft = 62 ,
  DoubleBarLeft = 63 , StressMarking = 64 , ResetBar = 65 , Superscript = 73 ,
  Subscript = 74 , ResetSuperscript = 75 , ResetSubscript = 75 }

Functions

- std::string color_bg (const Term::Color::Name &name)
- std::string color_bg (const std::uint8_t &value)
- std::string color_bg (const std::uint8_t &red, const std::uint8_t &green, const std::uint8_t &blue)
- std::string color_bg (const Color &color)
- std::string color_fg (const Term::Color::Name &name)
- std::string color_fg (const std::uint8_t &value)
- std::string color_fg (const std::uint8_t &red, const std::uint8_t &green, const std::uint8_t &blue)
- std::string color_fg (const Color &color)
- Term::Cursor cursor_position ()
- std::string cursor_move (const std::size_t &row, const std::size_t &column)
- std::string cursor_up (const std::size_t &rows)
- std::string cursor_down (const std::size_t &rows)
- std::string cursor_left (const std::size_t &columns)
- std::string cursor_right (const std::size_t &columns)
- std::string cursor_position_report ()
- std::string cursor_off ()
- std::string cursor_on ()
- std::string clear_eol ()
- Term::Event read_event ()
- constexpr bool operator== (Key l, MetaKey r)
- constexpr bool operator== (MetaKey l, Key r)
- constexpr bool operator< (MetaKey l, Key r)
- constexpr bool operator< (Key l, MetaKey r)
- constexpr bool operator!= (Key l, MetaKey r)
- constexpr bool operator!= (MetaKey l, Key r)
- constexpr bool operator>= (MetaKey l, Key r)
- constexpr bool operator>= (Key l, MetaKey r)
- constexpr bool operator> (MetaKey l, Key r)
- constexpr bool operator> (Key l, MetaKey r)
- constexpr bool operator<= (MetaKey l, Key r)
- constexpr bool operator<= (Key l, MetaKey r)
- constexpr Key operator+ (MetaKey metakey, Key key)
- constexpr Key operator+ (Key key, MetaKey meta)
- constexpr Key operator+ (MetaKey::Value l, Key r)
- constexpr Key operator+ (Key l, MetaKey::Value r)
- constexpr Key operator+ (MetaKey::Value l, Key::value_type r)
- constexpr Key operator+ (Key::value_type l, MetaKey::Value r)
- std::uint16_t returnCode () noexcept

- Result prompt (const std::string &message, const std::string &first_option, const std::string &second_option, const std::string &prompt_indicator, bool)

  A simple yes/no prompt, requires the user to press the ENTER key to continue.
- Result_simple prompt_simple (const std::string &message)

  The most simple prompt possible, requires the user to press enter to continue.
- std::string concat (const std::vector< std::string > &)
- std::vector< std::string > split (const std::string &)
- void print_left_curly_bracket (Term::Window &, const std::size_t &, const std::size_t &, const std::size_t &)
- void render (Term::Window &, const Model &, const std::size_t &)
- std::string prompt_multiline (const std::string &, std::vector< std::string > &, std::function< bool(std::string)> &)
- std::string clear_screen ()
- std::string screen_save ()
- std::string screen_load ()
- Screen screen_size ()
- std::string style (const Term::Style &style)
- template<class Stream >
  Stream & operator<< (Stream &stream, const Term::Style &style_type)
- Term::TOstream & operator<< (Term::TOstream &term, const Term::Style &style_type)
- std::string terminal_title (const std::string &title)
- std::string clear_buffer ()
- bool is_stdin_a_tty ()

  *Check if **stdin** is a **tty**.*
- bool is_stdout_a_tty ()

  *Check if **stdout** is a **tty**.*
- bool is_stderr_a_tty ()

  *Check if **stderr** is a **tty**.*
- std::string homepage () noexcept

  Homepage of cpp-terminal.

Variables

- TIstream & cin = reinterpret_cast<Term::TIstream&>(cin_buffer)
- TOstream & cout = reinterpret_cast<Term::TOstream&>(cout_buffer)
- TOstream & cerr = reinterpret_cast<Term::TOstream&>(cerr_buffer)
- TOstream & clog = reinterpret_cast<Term::TOstream&>(clog_buffer)
- Term::Terminal & terminal = reinterpret_cast<Term::Terminal&>(::terminal_buffer)

7.1.1 Enumeration Type Documentation

Option

```
enum class Term::Option :  std::int16_t  [strong]
```

Option to set-up the terminal.

Enumerator

Raw	Set terminal in **raw** mode.
Cooked	Set terminal in **cooked** mode.
ClearScreen	Clear the screen (and restore its states when the program stops).
NoClearScreen	Doesn't clear the screen.

Enumerator

SignalKeys	Enable the signal keys (Ctrl+C, etc...), if activated these keys will have their default OS behaviour.
NoSignalKeys	Disable the signal keys (Ctrl+C, etc...) will not be processed by the OS and will appears has standard combination keys.
Cursor	Show the cursor.
NoCursor	Hide the cursor (and restore its states when the program stops).

Definition at line 22 of file options.hpp.

```
00023 {
00024   Raw           = 1,
00025   Cooked        = -1,
00026   ClearScreen   = 2,
00027   NoClearScreen = -2,
00028   SignalKeys    = 3,
00029   NoSignalKeys  = -3,
00030   Cursor        = 4,
00031   NoCursor      = -4
00032 };
```

Result

```
enum class Term::Result  [strong]
```

Enumerator

Yes	Returned if the user chose **yes**.
No	Returned if the user chose **no**.
Error	Returned if no terminal is attached to the program.
None	Returned if the enter key was pressed without additional input.
Abort	Returned if CTRL+C was pressed.
Invalid	Returned if the given input did not match the case **yes** of **no**.

Definition at line 21 of file prompt.hpp.

```
00022 {
00023   Yes,
00024   No,
00025   Error,
00026   None,
00027   Abort,
00028   Invalid
00029 };
```

Result_simple

```
enum class Term::Result_simple  [strong]
```

Enumerator

Yes	Returned if the user chose **yes**.
No	Returned if the user chose no or invalid / no input or if no terminal is attached.
Abort	Returned if CTRL+C was pressed.

Definition at line 44 of file prompt.hpp.

```
00045 {
00046
00047   Yes,
00048   No,
00049   Abort
00050 };
```

Style

```
enum class Term::Style :  std::uint8_t  [strong]
```

Enumerator

Reset	resets all attributes (styles and colors)
Bold	Thick text font.
Dim	lighter, slimmer text font
Italic	slightly bend text font
Underline	draws a line below the text
Blink	Sets blinking to less than 150 times per minute.
BlinkRapid	MS-DOS ANSI.SYS, 150+ per minute; not widely supported.
Reversed	swap foreground and background colors
Conceal	Note: not widely supported.
Crossed	strikes through the text, mostly supported
Font0	Primary or default font.
ResetFont	
Font1	
Font2	
Font3	
Font4	
Font5	
Font6	
Font7	
Font8	
Font9	
Font10	Fraktur / Gothic font.
DoublyUnderlinedOrNotBold	
ResetBold	
ResetDim	
ResetItalic	
ResetUnderline	
ResetBlink	
ResetBlinkRapid	
ResetReversed	
ResetConceal	
ResetCrossed	
DefaultForegroundColor	
DefaultBackgroundColor	
Frame	
Encircle	
Overline	
ResetFrame	
ResetEncircle	
ResetOverline	

Enumerator

DefaultUnderlineColor	non standard, implemented in Kitty, VTE, mintty, and iTerm2
BarRight	draw a vertical bar on the right side of the character
DoubleBarRight	draw a double vertical bar to the right
BarLeft	draw a vertical bar on the left side of the character
DoubleBarLeft	draw a double vertical bar to the left
StressMarking	reset all bars left and right double and simple
ResetBar	
Superscript	
Subscript	
ResetSuperscript	
ResetSubscript	

Definition at line 23 of file style.hpp.

```
00024 {
00025
00026   Reset      = 0,
00027   Bold       = 1,
00028   Dim        = 2,
00029   Italic     = 3,
00030   Underline  = 4,
00031   Blink      = 5,
00032   BlinkRapid = 6,
00033   Reversed   = 7,
00034   Conceal    = 8,
00035   Crossed    = 9,
00036
00037   // different fonts
00038   Font0     = 10,
00039   ResetFont = 10,
00040   Font1     = 11,
00041   Font2     = 12,
00042   Font3     = 13,
00043   Font4     = 14,
00044   Font5     = 15,
00045   Font6     = 16,
00046   Font7     = 17,
00047   Font8     = 18,
00048   Font9     = 19,
00049   Font10    = 20,
00050
00051   // Double-underline per ECMA-48,[5] 8.3.117 but instead disables bold intensity on several
    terminals,
00052   // including in the Linux kernel's console before version 4.17
00053   DoublyUnderlinedOrNotBold = 21,
00054
00055   // resets corresponding styles
00056   ResetBold       = 22,
00057   ResetDim        = 22,
00058   ResetItalic     = 23,
00059   ResetUnderline  = 24,
00060   ResetBlink      = 25,
00061   ResetBlinkRapid = 25,
00062   ResetReversed   = 27,
00063   ResetConceal    = 28,
00064   ResetCrossed    = 29,
00065
00066   // sets the foreground and background color to the implementation defined colors
00067   DefaultForegroundColor = 39,
00068   DefaultBackgroundColor = 49,
00069
00070   Frame         = 51,
00071   Encircle      = 52,
00072   Overline      = 53,  // draw a line over the text, barely supported
00073   ResetFrame    = 54,
00074   ResetEncircle = 54,
00075   ResetOverline = 55,
00076
00077   // sets the underline color to the implementation defined colors
00078   DefaultUnderlineColor = 59,
00079
00080   BarRight       = 60,
00081   DoubleBarRight = 61,
00082   BarLeft        = 62,
00083   DoubleBarLeft  = 63,
```

```
00084   StressMarking  = 64,
00085
00086   ResetBar = 65,  // resets 60 - 64 inclusive
00087
00088   Superscript      = 73,  // only implemented in mintty
00089   Subscript        = 74,  // only implemented in mintty
00090   ResetSuperscript = 75,  // only implemented in mintty
00091   ResetSubscript   = 75
00092 };
```

7.1.2 Function Documentation

clear_buffer()

```
std::string Term::clear_buffer ( )
```

Definition at line 23 of file terminal.cpp.

```
00023 { return "\u001b[3J"; }
```

clear_eol()

```
std::string Term::clear_eol ( )
```

Definition at line 44 of file cursor.cpp.

```
00044 { return "\u001b[K"; }
```

clear_screen()

```
std::string Term::clear_screen ( )
```

Definition at line 20 of file screen.cpp.

```
00020 { return "\u001b[2J"; }
```

color_bg() [1/4]

```
std::string Term::color_bg (
            const Color & color )
```

Definition at line 87 of file color.cpp.

```
00088 {
00089   if(color.getType() == Term::Color::Type::Unset || color.getType() == Term::Color::Type::NoColor)
      return "";
00090   switch(Term::Terminfo::getColorMode())
00091   {
00092     case Term::Terminfo::ColorMode::Unset:
00093     case Term::Terminfo::ColorMode::NoColor: return {};
00094     case Term::Terminfo::ColorMode::Bit3: return "\u001b[" +
      std::to_string(static_cast<uint8_t>(color.to3bits()) + 40) + "m\u001b[K";
00095     case Term::Terminfo::ColorMode::Bit4: return "\u001b[" +
      std::to_string(static_cast<uint8_t>(color.to4bits()) + 40) + "m\u001b[K";
00096     case Term::Terminfo::ColorMode::Bit8:
00097       if(color.getType() == Term::Color::Type::Bit4 || color.getType() == Term::Color::Type::Bit3)
      return "\u001b[" + std::to_string(static_cast<uint8_t>(color.to4bits()) + 40) + "m\u001b[K";
00098       else
00099         return "\u001b[48;5;" + std::to_string(color.to8bits()) + "m\u001b[K";
00100     case Term::Terminfo::ColorMode::Bit24:
00101       if(color.getType() == Term::Color::Type::Bit3 || color.getType() == Term::Color::Type::Bit4)
      return "\u001b[" + std::to_string(static_cast<uint8_t>(color.to4bits()) + 40) + "m\u001b[K";
00102       else if(color.getType() == Term::Color::Type::Bit8)
00103         return "\u001b[48;5;" + std::to_string(color.to8bits()) + "m\u001b[K";
00104       else
00105         return "\u001b[48;2;" + std::to_string(color.to24bits()[0]) + ';' +
      std::to_string(color.to24bits()[1]) + ';' + std::to_string(color.to24bits()[2]) + "m\u001b[K";
00106     default: return {};
00107   }
00108 }
```

color_bg() [2/4]

```
std::string Term::color_bg (
              const std::uint8_t & red,
              const std::uint8_t & green,
              const std::uint8_t & blue )
```

Definition at line 83 of file color.cpp.

```
00083 { return color_bg(Color(r, g, b)); }
```

color_bg() [3/4]

```
std::string Term::color_bg (
              const std::uint8_t & value )
```

Definition at line 81 of file color.cpp.

```
00081 { return color_bg(Color(color)); }
```

color_bg() [4/4]

```
std::string Term::color_bg (
              const Term::Color::Name & name )
```

Definition at line 79 of file color.cpp.

```
00079 { return color_bg(Color(color)); }
```

color_fg() [1/4]

```
std::string Term::color_fg (
              const Color & color )
```

Definition at line 116 of file color.cpp.

```
00117 {
00118   if(color.getType() == Term::Color::Type::Unset || color.getType() == Term::Color::Type::NoColor) {
      return {}; }
00119   switch(Term::Terminfo::getColorMode())
00120   {
00121     case Term::Terminfo::ColorMode::Unset:
00122     case Term::Terminfo::ColorMode::NoColor: return "";
00123     case Term::Terminfo::ColorMode::Bit3: return "\u001b[" +
      std::to_string(static_cast<uint8_t>(color.to3bits()) + 30) + "m";
00124     case Term::Terminfo::ColorMode::Bit4: return "\u001b[" +
      std::to_string(static_cast<uint8_t>(color.to4bits()) + 30) + "m";
00125     case Term::Terminfo::ColorMode::Bit8:
00126       if(color.getType() == Term::Color::Type::Bit4 || color.getType() == Term::Color::Type::Bit3)
      return "\u001b[" + std::to_string(static_cast<uint8_t>(color.to4bits()) + 30) + "m";
00127       else
00128         return "\u001b[38;5;" + std::to_string(color.to8bits()) + "m";
00129     case Term::Terminfo::ColorMode::Bit24:
00130       if(color.getType() == Term::Color::Type::Bit3 || color.getType() == Term::Color::Type::Bit4)
      return "\u001b[" + std::to_string(static_cast<uint8_t>(color.to4bits()) + 30) + "m";
00131       else if(color.getType() == Term::Color::Type::Bit8)
00132         return "\u001b[38;5;" + std::to_string(color.to8bits()) + "m";
00133       else
00134         return "\u001b[38;2;" + std::to_string(color.to24bits()[0]) + ';' +
      std::to_string(color.to24bits()[1]) + ';' + std::to_string(color.to24bits()[2]) + "m";
00135     default: return {};
00136   }
00137 }
```

color_fg() [2/4]

```
std::string Term::color_fg (
            const std::uint8_t & red,
            const std::uint8_t & green,
            const std::uint8_t & blue )
```

Definition at line 114 of file color.cpp.

```
00114 { return color_fg(Color(red, green, blue)); }
```

color_fg() [3/4]

```
std::string Term::color_fg (
            const std::uint8_t & value )
```

Definition at line 112 of file color.cpp.

```
00112 { return color_fg(Color(value)); }
```

color_fg() [4/4]

```
std::string Term::color_fg (
            const Term::Color::Name & name )
```

Definition at line 110 of file color.cpp.

```
00110 { return color_fg(Color(name)); }
```

concat()

```
std::string Term::concat (
            const std::vector< std::string > & lines )
```

Definition at line 143 of file prompt.cpp.

```
00144 {
00145   std::string s;
00146   for(auto& line: lines) { s.append(line + "\n"); }
00147   return s;
00148 }
```

cursor_down()

```
std::string Term::cursor_down (
            const std::size_t & rows )
```

Definition at line 36 of file cursor.cpp.

```
00036 { return "\u001b[" + std::to_string(rows) + 'B'; }
```

cursor_left()

```
std::string Term::cursor_left (
            const std::size_t & columns )
```

Definition at line 40 of file cursor.cpp.

```
00040 { return "\u001b[" + std::to_string(columns) + 'D'; }
```

cursor_move()

```
std::string Term::cursor_move (
            const std::size_t & row,
            const std::size_t & column )
```

Definition at line 32 of file cursor.cpp.

```
00032 { return "\u001b[" + std::to_string(row) + ';' + std::to_string(column) + 'H'; }
```

cursor_off()

```
std::string Term::cursor_off ( )
```

Definition at line 28 of file cursor.cpp.

```
00028 { return "\u001b[?25l"; }
```

cursor_on()

```
std::string Term::cursor_on ( )
```

Definition at line 30 of file cursor.cpp.

```
00030 { return "\u001b[?25h"; }
```

cursor_position()

```
Term::Cursor Term::cursor_position ( )
```

Definition at line 23 of file cursor.cpp.

```
00024 {
00025   static const Term::Private::FileInitializer files_init;
00026   if(Term::Private::in.null()) { return {}; }
00027 #if defined(_WIN32)
00028   CONSOLE_SCREEN_BUFFER_INFO inf;
00029   if(GetConsoleScreenBufferInfo(Private::out.handle(), &inf)) return
      Term::Cursor(static_cast<std::size_t>(inf.dwCursorPosition.Y + 1),
      static_cast<std::size_t>(inf.dwCursorPosition.X + 1));
00030   else
00031     return Term::Cursor(0, 0);
00032 #else
00033   std::string ret;
00034   std::size_t nread{0};
00035   Term::Private::in.lockIO();
00036   // Hack to be sure we can do this all the time "Cooked" or "Raw" mode
00037   ::termios actual;
00038   if(!Private::out.null())
00039   {
00040     if(tcgetattr(Private::out.fd(), &actual) == -1) { return {}; }
00041   }
00042   ::termios raw = actual;
00043   // Put terminal in raw mode
00044   raw.c_iflag &= ~(BRKINT | ICRNL | INPCK | ISTRIP | IXON);
00045   // This disables output post-processing, requiring explicit \r\n. We
00046   // keep it enabled, so that in C++, one can still just use std::endl
00047   // for EOL instead of "\r\n".
00048   // raw.c_oflag &= ~(OPOST);
00049   raw.c_lflag &= ~(ECHO | ICANON | IEXTEN);
00050   raw.c_lflag &= ~ISIG;
00051   raw.c_cc[VMIN]  = 1;
00052   raw.c_cc[VTIME] = 0;
00053   if(!Private::out.null()) tcsetattr(Private::out.fd(), TCSAFLUSH, &raw);
00054   Term::Private::out.write(Term::cursor_position_report());
00055   while(nread == 0) { ::ioctl(Private::in.fd(), FIONREAD, &nread); }
00056   ret = Term::Private::in.read();
00057   tcsetattr(Private::out.fd(), TCSAFLUSH, &actual);
00058   Term::Private::in.unlockIO();
00059   try
00060   {
```

```
00061     if(ret[0] == '\033' && ret[1] == '[' && ret[ret.size() - 1] == 'R')
00062     {
00063       std::size_t found = ret.find(';', 2);
00064       if(found != std::string::npos) { return Cursor(std::stoi(ret.substr(2, found - 2)),
      std::stoi(ret.substr(found + 1, ret.size() - (found + 2)))); }
00065       return {};
00066     }
00067     return {};
00068   }
00069   catch(...)
00070   {
00071     return {};
00072   }
00073 #endif
00074 }
```

cursor_position_report()

```
std::string Term::cursor_position_report ( )
```

Definition at line 42 of file cursor.cpp.

```
00042 { return "\u001b[6n"; }
```

cursor_right()

```
std::string Term::cursor_right (
              const std::size_t & columns )
```

Definition at line 38 of file cursor.cpp.

```
00038 { return "\u001b[" + std::to_string(columns) + 'C'; }
```

cursor_up()

```
std::string Term::cursor_up (
              const std::size_t & rows )
```

Definition at line 34 of file cursor.cpp.

```
00034 { return "\u001b[" + std::to_string(rows) + 'A'; }
```

homepage()

```
std::string Term::homepage ( )  [noexcept]
```

Homepage of cpp-terminal.

Returns

std::string return the URL of the cpp-terminal project.

is_stderr_a_tty()

```
bool Term::is_stderr_a_tty ( )
```

Check if **stderr** is a **tty**.

Returns

true : **stderr** is a **tty**.

false : **stderr** is not a **tty**.

Definition at line 36 of file tty.cpp.
```
00036 { return ::is_a_tty(stderr); }
```

is_stdin_a_tty()

```
bool Term::is_stdin_a_tty ( )
```

Check if **stdin** is a **tty**.

Returns

true : **stdin** is a **tty**.

false : **stdin** is not a **tty**.

Definition at line 32 of file tty.cpp.
```
00032 { return ::is_a_tty(stdin); }
```

is_stdout_a_tty()

```
bool Term::is_stdout_a_tty ( )
```

Check if **stdout** is a **tty**.

Returns

true : **stdout** is a **tty**.

false : **stdout** is not a **tty**.

Definition at line 34 of file tty.cpp.
```
00034 { return ::is_a_tty(stdout); }
```

operator"!=() [1/2]

```
constexpr bool Term::operator!= (
            Key l,
            MetaKey r )  [constexpr]
```

Definition at line 430 of file key.hpp.
```
00430 { return static_cast<std::int32_t>(l) != static_cast<std::int32_t>(r); }
```

operator"!=() [2/2]

```
constexpr bool Term::operator!= (
            MetaKey l,
            Key r )  [constexpr]
```

Definition at line 431 of file key.hpp.

```
00431 { return static_cast<std::int32_t>(l) != static_cast<std::int32_t>(r); }
```

operator+() [1/6]

```
constexpr Key Term::operator+ (
            Key key,
            MetaKey meta )  [constexpr]
```

Definition at line 443 of file key.hpp.

```
00443 { return meta + key; }
```

operator+() [2/6]

```
constexpr Key Term::operator+ (
            Key l,
            MetaKey::Value r )  [constexpr]
```

Definition at line 446 of file key.hpp.

```
00446 { return l + MetaKey(r); }
```

operator+() [3/6]

```
constexpr Key Term::operator+ (
            Key::value_type l,
            MetaKey::Value r )  [constexpr]
```

Definition at line 448 of file key.hpp.

```
00448 { return Key(l) + MetaKey(r); }
```

operator+() [4/6]

```
constexpr Key Term::operator+ (
            MetaKey metakey,
            Key key )  [constexpr]
```

Definition at line 442 of file key.hpp.

```
00442 { return Key(key.value + ((metakey == MetaKey::Value::Ctrl && !key.hasCtrlAll() && !key.empty()) ?
      static_cast<std::int32_t>(MetaKey::Value::Ctrl) : 0) + ((metakey == MetaKey::Value::Alt &&
      !key.hasAlt() && !key.empty()) ? static_cast<std::int32_t>(MetaKey::Value::Alt) : 0)); }
```

operator+() [5/6]

```
constexpr Key Term::operator+ (
            MetaKey::Value l,
            Key r )  [constexpr]
```

Definition at line 445 of file key.hpp.

```
00445 { return MetaKey(l) + r; }
```

operator+() [6/6]

```
constexpr Key Term::operator+ (
            MetaKey::Value l,
            Key::value_type r )  [constexpr]
```

Definition at line 447 of file key.hpp.

```
00447 { return MetaKey(l) + Key(r); }
```

operator<() [1/2]

```
constexpr bool Term::operator< (
            Key l,
            MetaKey r )  [constexpr]
```

Definition at line 428 of file key.hpp.

```
00428 { return static_cast<std::int32_t>(l) < static_cast<std::int32_t>(r); }
```

operator<() [2/2]

```
constexpr bool Term::operator< (
            MetaKey l,
            Key r )  [constexpr]
```

Definition at line 427 of file key.hpp.

```
00427 { return static_cast<std::int32_t>(l) < static_cast<std::int32_t>(r); }
```

operator<<() [1/2]

```
template<class Stream >
Stream & Term::operator<< (
            Stream & stream,
            const Term::Style & style_type )
```

Definition at line 96 of file style.hpp.

```
00096 { return stream << style(style_type); }
```

operator<<() [2/2]

```
Term::TOstream & Term::operator<< (
            Term::TOstream & term,
            const Term::Style & style_type )  [inline]
```

Definition at line 98 of file style.hpp.

```
00098 { return term << style(style_type); }
```

operator<=() [1/2]

```
constexpr bool Term::operator<= (
            Key l,
            MetaKey r )  [constexpr]
```

Definition at line 440 of file key.hpp.

```
00440 { return static_cast<std::int32_t>(l) <= static_cast<std::int32_t>(r); }
```

operator<=() [2/2]

```
constexpr bool Term::operator<= (
            MetaKey l,
            Key r )  [constexpr]
```

Definition at line 439 of file key.hpp.

```
00439 { return static_cast<std::int32_t>(l) <= static_cast<std::int32_t>(r); }
```

operator==() [1/2]

```
constexpr bool Term::operator== (
            Key l,
            MetaKey r )  [constexpr]
```

Definition at line 424 of file key.hpp.

```
00424 { return static_cast<std::int32_t>(l) == static_cast<std::int32_t>(r); }
```

operator==() [2/2]

```
constexpr bool Term::operator== (
            MetaKey l,
            Key r )  [constexpr]
```

Definition at line 425 of file key.hpp.

```
00425 { return static_cast<std::int32_t>(l) == static_cast<std::int32_t>(r); }
```

operator>() [1/2]

```
constexpr bool Term::operator> (
            Key l,
            MetaKey r )  [constexpr]
```

Definition at line 437 of file key.hpp.

```
00437 { return static_cast<std::int32_t>(l) > static_cast<std::int32_t>(r); }
```

operator>() [2/2]

```
constexpr bool Term::operator> (
            MetaKey l,
            Key r )  [constexpr]
```

Definition at line 436 of file key.hpp.

```
00436 { return static_cast<std::int32_t>(l) > static_cast<std::int32_t>(r); }
```

operator>=() [1/2]

```
constexpr bool Term::operator>= (
            Key l,
            MetaKey r )  [constexpr]
```

Definition at line 434 of file key.hpp.

```
00434 { return static_cast<std::int32_t>(l) >= static_cast<std::int32_t>(r); }
```

operator>=() [2/2]

```
constexpr bool Term::operator>= (
            MetaKey l,
            Key r )  [constexpr]
```

Definition at line 433 of file key.hpp.

```
00433 { return static_cast<std::int32_t>(l) >= static_cast<std::int32_t>(r); }
```

print_left_curly_bracket()

```
void Term::print_left_curly_bracket (
            Term::Window & scr,
            const std::size_t & x,
            const std::size_t & y1,
            const std::size_t & y2 )
```

Definition at line 175 of file prompt.cpp.

```
00176 {
00177   std::size_t h{y2 - y1 + 1};
00178   if(h == 1) { scr.set_char(x, y1, UU("]")); }
00179   else
00180   {
00181     scr.set_char(x, y1, UU("┐"));
00182     for(std::size_t j = y1 + 1; j <= y2 - 1; j++) { scr.set_char(x, j, UU("│")); }
00183     scr.set_char(x, y2, UU("┘"));
00184   }
00185 }
```

prompt()

```
Term::Result Term::prompt (
            const std::string & message,
            const std::string & first_option,
            const std::string & second_option,
            const std::string & prompt_indicator,
            bool immediate )
```

A simple yes/no prompt, requires the user to press the ENTER key to continue.

The arguments are used like this: 1 [2/3]4 <user Input>. The immediate switch indicates toggles whether pressing enter for confirming the input is required or not.

Parameters

message	
first_option	
second_option	
prompt_indicator	

Returns

Result

Definition at line 26 of file prompt.cpp.

```
00027 {
00028   Term::terminal.setOptions(Option::NoClearScreen, Option::NoSignalKeys, Option::Cursor,
    Term::Option::Raw);
00029   std::cout « message « " [" « first_option « '/' « second_option « ']' « prompt_indicator « ' ' «
    std::flush;
00030
00031   if(!Term::is_stdin_a_tty())
00032   {
00033     std::cout « '\n' « std::flush;
00034     return Result::Error;
00035   }
00036
00037   Term::Key key;
00038
00039   if(immediate)
00040   {
00041     while(true)
00042     {
00043       key = Term::read_event();
00044       if(key == Term::Key::NoKey) continue;
00045       if(key == Term::Key::y || key == Term::Key::Y)
00046       {
00047         std::cout « '\n' « std::flush;
00048         return Result::Yes;
00049       }
00050       else if(key == Term::Key::n || key == Term::Key::N)
00051       {
00052         std::cout « '\n' « std::flush;
00053         return Result::No;
00054       }
00055       else if(key == Term::Key::Ctrl_C || key == Term::Key::Ctrl_D)
00056       {
00057         std::cout « '\n' « std::flush;
00058         return Result::Abort;
00059       }
00060       else if(key == Term::Key::Enter)
00061       {
00062         std::cout « '\n' « std::flush;
00063         return Result::None;
00064       }
00065       else
00066       {
00067         std::cout « '\n' « std::flush;
00068         return Result::Invalid;
00069       }
00070     }
00071   }
00072   else
00073   {
00074     std::string input;
00075     while(true)
00076     {
00077       key = Term::read_event();
00078       if(key == Term::Key::NoKey) continue;
00079       if(key >= 'a' && key <= 'z')
00080       {
00081         std::cout « (char)key « std::flush;
00082         input.push_back(static_cast<char>(key));
00083       }
00084       else if(key >= 'A' && key <= 'Z')
00085       {
00086         std::cout « (char)key « std::flush;
00087         input.push_back(static_cast<char>(key.tolower()));  // convert upper case to lowercase
00088       }
00089       else if(key == Term::Key::Ctrl_C || key == Term::Key::Ctrl_D)
00090       {
00091         std::cout « '\n';
00092         return Result::Abort;
00093       }
00094       else if(key == Term::Key::Backspace)
00095       {
00096         if(input.empty() != 0)
00097         {
00098           std::cout « "\u001b[D \u001b[D" « std::flush;  // erase last line and move the cursor back
00099           input.pop_back();
00100         }
00101       }
00102       else if(key == Term::Key::Enter)
00103       {
00104         if(input == "y" || input == "yes")
00105         {
00106           std::cout « '\n' « std::flush;
00107           return Result::Yes;
00108         }
00109         else if(input == "n" || input == "no")
00110         {
00111           std::cout « '\n' « std::flush;
```

```
00112           return Result::No;
00113         }
00114         else if(input.empty())
00115         {
00116           std::cout « '\n' « std::flush;
00117           return Result::None;
00118         }
00119         else
00120         {
00121           std::cout « '\n' « std::flush;
00122           return Result::Invalid;
00123         }
00124       }
00125     }
00126   }
00127 }
```

prompt_multiline()

```
std::string Term::prompt_multiline (
              const std::string & prompt_string,
              std::vector< std::string > & m_history,
              std::function< bool(std::string)> & iscomplete )
```

Definition at line 209 of file prompt.cpp.

```
00210 {
00211   Term::Cursor cursor;
00212   Term::Screen screen(25, 80);
00213   bool         term_attached = Term::is_stdin_a_tty();
00214   if(is_stdin_a_tty())
00215   {
00216     cursor = cursor_position();
00217     screen = screen_size();
00218   }
00219
00220   Model m;
00221   m.prompt_string = prompt_string;
00222
00223   // Make a local copy of history that can be modified by the user. All
00224   // changes will be forgotten once a command is submitted.
00225   std::vector<std::string> history     = m_history;
00226   std::size_t              history_pos = history.size();
00227   history.push_back(concat(m.lines));  // Push back empty input
00228
00229   Term::Window scr(screen.columns(), 1);
00230   Term::Key    key;
00231   render(scr, m, screen.columns());
00232   std::cout « scr.render(1, cursor.row(), term_attached) « std::flush;
00233   bool not_complete = true;
00234   while(not_complete)
00235   {
00236     key = Term::read_event();
00237     if(key == Term::Key::NoKey) continue;
00238     if(key.isprint())
00239     {
00240       std::string before = m.lines[m.cursor_row - 1].substr(0, m.cursor_col - 1);
00241       std::string newchar;
00242       newchar.push_back(static_cast<char>(key));
00243       std::string after         = m.lines[m.cursor_row - 1].substr(m.cursor_col - 1);
00244       m.lines[m.cursor_row - 1] = before += newchar += after;
00245       m.cursor_col++;
00246     }
00247     else if(key == Key::Ctrl_D)
00248     {
00249       if(m.lines.size() == 1 && m.lines[m.cursor_row - 1].empty())
00250       {
00251         m.lines[m.cursor_row - 1].push_back(static_cast<char>(Key::Ctrl_D));
00252         std::cout « "\n" « std::flush;
00253         m_history.push_back(m.lines[0]);
00254         return m.lines[0];
00255       }
00256     }
00257     else
00258     {
00259       switch(key)
00260       {
00261         case Key::Enter:
00262           not_complete = !iscomplete(concat(m.lines));
00263           if(not_complete) key = Key(static_cast<Term::Key>(Term::MetaKey::Value::Alt +
      Term::Key::Enter));
```

```
00264           else
00265             break;
00266           CPP_TERMINAL_FALLTHROUGH;
00267         case Key::Backspace:
00268           if(m.cursor_col > 1)
00269           {
00270             std::string before        = m.lines[m.cursor_row - 1].substr(0, m.cursor_col - 2);
00271             std::string after         = m.lines[m.cursor_row - 1].substr(m.cursor_col - 1);
00272             m.lines[m.cursor_row - 1] = before + after;
00273             m.cursor_col--;
00274           }
00275           else if(m.cursor_col == 1 && m.cursor_row > 1)
00276           {
00277             m.cursor_col = m.lines[m.cursor_row - 2].size() + 1;
00278             m.lines[m.cursor_row - 2] += m.lines[m.cursor_row - 1];
00279             m.lines.erase(m.lines.begin() + static_cast<long>(m.cursor_row) - 1);
00280             m.cursor_row--;
00281           }
00282           break;
00283         case Key::Del:
00284           if(m.cursor_col <= m.lines[m.cursor_row - 1].size())
00285           {
00286             std::string before        = m.lines[m.cursor_row - 1].substr(0, m.cursor_col - 1);
00287             std::string after         = m.lines[m.cursor_row - 1].substr(m.cursor_col);
00288             m.lines[m.cursor_row - 1] = before + after;
00289           }
00290           break;
00291         case Key::ArrowLeft:
00292           if(m.cursor_col > 1) { m.cursor_col--; }
00293           break;
00294         case Key::ArrowRight:
00295           if(m.cursor_col <= m.lines[m.cursor_row - 1].size()) { m.cursor_col++; }
00296           break;
00297         case Key::Home: m.cursor_col = 1; break;
00298         case Key::End: m.cursor_col = m.lines[m.cursor_row - 1].size() + 1; break;
00299         case Key::ArrowUp:
00300           if(m.cursor_row == 1)
00301           {
00302             if(history_pos > 0)
00303             {
00304               history[history_pos] = concat(m.lines);
00305               history_pos--;
00306               m.lines      = split(history[history_pos]);
00307               m.cursor_row = m.lines.size();
00308               if(m.cursor_col > m.lines[m.cursor_row - 1].size() + 1) { m.cursor_col =
      m.lines[m.cursor_row - 1].size() + 1; }
00309               if(m.lines.size() > scr.get_h()) { scr.set_h(m.lines.size()); }
00310             }
00311           }
00312           else
00313           {
00314             m.cursor_row--;
00315             if(m.cursor_col > m.lines[m.cursor_row - 1].size() + 1) { m.cursor_col =
      m.lines[m.cursor_row - 1].size() + 1; }
00316           }
00317           break;
00318         case Key::ArrowDown:
00319           if(m.cursor_row == m.lines.size())
00320           {
00321             if(history_pos < history.size() - 1)
00322             {
00323               history[history_pos] = concat(m.lines);
00324               history_pos++;
00325               m.lines      = split(history[history_pos]);
00326               m.cursor_row = 1;
00327               if(m.cursor_col > m.lines[m.cursor_row - 1].size() + 1) { m.cursor_col =
      m.lines[m.cursor_row - 1].size() + 1; }
00328               if(m.lines.size() > scr.get_h()) { scr.set_h(m.lines.size()); }
00329             }
00330           }
00331           else
00332           {
00333             m.cursor_row++;
00334             if(m.cursor_col > m.lines[m.cursor_row - 1].size() + 1) { m.cursor_col =
      m.lines[m.cursor_row - 1].size() + 1; }
00335           }
00336           break;
00337         case Key::Ctrl_N:
00338         {
00339           std::string before        = m.lines[m.cursor_row - 1].substr(0, m.cursor_col - 1);
00340           std::string after         = m.lines[m.cursor_row - 1].substr(m.cursor_col - 1);
00341           m.lines[m.cursor_row - 1] = before;
00342           if(m.cursor_row < m.lines.size())
00343           {
00344             // Not at the bottom row, can't push back
00345             m.lines.insert(m.lines.begin() + static_cast<long>(m.cursor_row), after);
00346           }
```

```
00347           else { m.lines.push_back(after); }
00348           m.cursor_col = 1;
00349           m.cursor_row++;
00350           if(m.lines.size() > scr.get_h()) { scr.set_h(m.lines.size()); }
00351           break;
00352         }
00353         default: break;
00354       }
00355     }
00356     render(scr, m, screen.columns());
00357     std::cout « scr.render(1, cursor.row(), term_attached) « std::flush;
00358     if(cursor.row() + (int)scr.get_h() - 1 > screen.rows())
00359     {
00360       cursor.setRow(static_cast<std::uint16_t>(static_cast<long>(screen.rows()) -
      (static_cast<long>(scr.get_h()) - 1)));
00361       std::cout « scr.render(1, cursor.row(), term_attached) « std::flush;
00362     }
00363   }
00364   std::string line_skips;
00365   for(std::size_t i = 0; i <= m.lines.size() - m.cursor_row; i++) { line_skips += "\n"; }
00366   std::cout « line_skips « std::flush;
00367   m_history.push_back(concat(m.lines));
00368   return concat(m.lines);
00369 }
```

prompt_simple()

```
Term::Result_simple Term::prompt_simple (
            const std::string & message )
```

The most simple prompt possible, requires the user to press enter to continue.

The arguments are used like this: 1 [y/N]: Invalid input, errors (like no attached terminal) all result in **no** as default.

Parameters

message	

Returns

Result_simple

Definition at line 129 of file prompt.cpp.

```
00130 {
00131   switch(prompt(message, "y", "N", ":", false))
00132   {
00133     case Result::Yes: return Result_simple::Yes;
00134     case Result::Abort: return Result_simple::Abort;
00135     case Result::No:     // falls through
00136     case Result::Error:  // falls through
00137     case Result::None:   // falls through
00138     case Result::Invalid:
00139     default: return Result_simple::No;
00140   }
00141 }
```

read_event()

```
Term::Event Term::read_event ( )
```

Definition at line 329 of file input.cpp.

```
00330 {
00331   m_input.startReading();
00332   return m_input.getEventBlocking();
00333 }
```

render()

```
void Term::render (
              Term::Window & scr,
              const Model & m,
              const std::size_t & cols )
```

Definition at line 187 of file prompt.cpp.

```
00188 {
00189   scr.clear();
00190   print_left_curly_bracket(scr, cols, 1, m.lines.size());
00191   scr.print_str(cols - 6, m.lines.size(), std::to_string(m.cursor_row) + "," +
      std::to_string(m.cursor_col));
00192   for(std::size_t j = 0; j < m.lines.size(); j++)
00193   {
00194     if(j == 0)
00195     {
00196       scr.fill_fg(1, j + 1, m.prompt_string.size(), m.lines.size(), Term::Color::Name::Green);
00197       scr.fill_style(1, j + 1, m.prompt_string.size(), m.lines.size(), Term::Style::Bold);
00198       scr.print_str(1, j + 1, m.prompt_string);
00199     }
00200     else
00201     {
00202       for(std::size_t i = 0; i < m.prompt_string.size() - 1; i++) { scr.set_char(i + 1, j + 1, '.'); }
00203     }
00204     scr.print_str(m.prompt_string.size() + 1, j + 1, m.lines[j]);
00205   }
00206   scr.set_cursor_pos(m.prompt_string.size() + m.cursor_col, m.cursor_row);
00207 }
```

returnCode()

```
std::uint16_t Term::returnCode ( )  [noexcept]
```

Definition at line 18 of file return_code.cpp.

```
00019 {
00020   static std::uint16_t               code{EXIT_FAILURE};
00021   const std::pair<bool, std::string> returnCode{Private::getenv("CPP_TERMINAL_BADSTATE")};
00022   try
00023   {
00024     if(returnCode.first && (std::stoi(returnCode.second) != EXIT_SUCCESS)) { code =
      static_cast<std::uint16_t>(std::stoi(returnCode.second)); }
00025   }
00026   catch(...)
00027   {
00028     code = EXIT_FAILURE;
00029   }
00030   return code;
00031 }
```

screen_load()

```
std::string Term::screen_load ( )
```

Definition at line 27 of file screen.cpp.

```
00028 {
00029   return "\u001b[?1049l\u001b8";  // restores screen, restore current cursor position FIXME
00030 }
```

screen_save()

```
std::string Term::screen_save ( )
```

Definition at line 22 of file screen.cpp.

```
00023 {
00024   return "\u001b7\u001b[?1049h";  // save current cursor position, save screen FIXME
00025 }
```

screen_size()

```
Term::Screen Term::screen_size ( )
```

Definition at line 20 of file screen.cpp.

```
00021 {
00022 #ifdef _WIN32
00023   CONSOLE_SCREEN_BUFFER_INFO inf;
00024   if(GetConsoleScreenBufferInfo(Private::out.handle(), &inf)) return
      Term::Screen(static_cast<std::size_t>(inf.srWindow.Bottom - inf.srWindow.Top + 1),
      static_cast<std::size_t>(inf.srWindow.Right - inf.srWindow.Left + 1));
00025   return Term::Screen();
00026 #else
00027   Term::Screen ret;
00028   struct winsize window
00029   {
00030     0, 0, 0, 0
00031   };
00032   if(ioctl(Private::out.fd(), TIOCGWINSZ, &window) != -1) ret = {window.ws_row, window.ws_col};
00033   return ret;
00034 #endif
00035 }
```

split()

```
std::vector< std::string > Term::split (
              const std::string & s )
```

Definition at line 150 of file prompt.cpp.

```
00151 {
00152   std::size_t              j = 0;
00153   std::vector<std::string> lines;
00154   lines.emplace_back("");
00155   if(s[s.size() - 1] != '\n') throw Term::Exception("\\n is required");
00156   for(std::size_t i = 0; i < s.size() - 1; i++)
00157   {
00158     if(s[i] == '\n')
00159     {
00160       j++;
00161       lines.emplace_back("");
00162     }
00163     else { lines[j].push_back(s[i]); }
00164   }
00165   return lines;
00166 }
```

style()

```
std::string Term::style (
              const Term::Style & style )
```

Definition at line 12 of file style.cpp.

```
00013 {
00014
      //https://unix.stackexchange.com/questions/212933/background-color-whitespace-when-end-of-the-terminal-reached
00015   std::string ret{"\u001b[" + std::to_string(static_cast<std::uint8_t>(style)) + 'm'};
00016   if(style == Term::Style::DefaultBackgroundColor) { ret += "\u001b[K"; }
00017   return ret;
00018 }
```

terminal_title()

```
std::string Term::terminal_title (
              const std::string & title )
```

Definition at line 21 of file terminal.cpp.

```
00021 { return "\u001b]0;" + title + '\a'; }
```

7.1.3 Variable Documentation

cerr

```
Term::TOstream & Term::cerr = reinterpret_cast<Term::TOstream&>(cerr_buffer)  [extern]
```

Definition at line 24 of file iostream.cpp.

cin

```
Term::TIstream & Term::cin = reinterpret_cast<Term::TIstream&>(cin_buffer)  [extern]
```

Definition at line 25 of file iostream.cpp.

clog

```
Term::TOstream & Term::clog = reinterpret_cast<Term::TOstream&>(clog_buffer)  [extern]
```

Definition at line 23 of file iostream.cpp.

cout

```
Term::TOstream & Term::cout = reinterpret_cast<Term::TOstream&>(cout_buffer)  [extern]
```

Definition at line 22 of file iostream.cpp.

terminal

```
Term::Terminal & Term::terminal = reinterpret_cast<Term::Terminal&>(::terminal_buffer)  [extern]
```

Definition at line 19 of file terminal.cpp.

7.2 Term::Private Namespace Reference

Classes

- class BlockingQueue
- class Errno
- class ErrnoException
- class FileHandler
- class FileInitializer
- class Input
- class InputFileHandler
- class OutputFileHandler
- class Sigwinch
- class WindowsError
- class WindowsException

Enumerations

- enum class ExceptionDestination : std::uint8_t { MessageBox = 0 , StdErr }

Functions

- std::uint8_t utf8_decode_step (std::uint8_t state, std::uint8_t octet, std::uint32_t *cpp)
- std::u32string utf8_to_utf32 (const std::string &str)
- bool is_valid_utf8_code_unit (const std::string &str)
- std::pair< bool, std::string > getenv (const std::string &env)

  Value of an environment variables.
- void ExceptionHandler (const ExceptionDestination &destination=ExceptionDestination::StdErr) noexcept
- std::string ask (const std::string &str)
- std::string to_narrow (const std::wstring &wstr)
- std::wstring to_wide (const std::string &str)
- std::string utf32_to_utf8 (const char32_t &codepoint, const bool &exception=false)

  Encode a codepoint using UTF-8 ***std::string*** .
- std::string utf32_to_utf8 (const std::u32string &str, const bool &exception=false)

  Encode a ***std::u32string*** *into UTF-8* ***std::string*** .

Variables

- InputFileHandler & in = reinterpret_cast<Term::Private::InputFileHandler&>(stdin_buffer)
- OutputFileHandler & out = reinterpret_cast<Term::Private::OutputFileHandler&>(stdout_buffer)
- volatile std::sig_atomic_t m_signalStatus {0}

7.2.1 Enumeration Type Documentation

ExceptionDestination

```
enum class Term::Private::ExceptionDestination :  std::uint8_t  [strong]
```

Enumerator

MessageBox	
StdErr	

Definition at line 87 of file exception.hpp.

```
00088 {
00089   MessageBox = 0,
00090   StdErr,
00091 };
```

7.2.2 Function Documentation

ask()

```
std::string Term::Private::ask (
            const std::string & str )
```

Definition at line 167 of file file.cpp.

```
00168 {
00169   Term::Private::out.write(str);
00170   std::string ret{Term::Private::in.read()};
00171   for(std::size_t i = 0; i != ret.size(); ++i) { Term::Private::out.write("\b \b"); }
00172   return ret;
00173 }
```

ExceptionHandler()

```
void Term::Private::ExceptionHandler (
            const ExceptionDestination & destination = ExceptionDestination::StdErr )  [noexcept]
```

Definition at line 171 of file exception.cpp.

```
00172 {
00173   try
00174   {
00175     std::exception_ptr exception{std::current_exception()};
00176     if(exception != nullptr) { std::rethrow_exception(exception); }
00177   }
00178   catch(const Term::Exception& exception)
00179   {
00180     switch(destination)
00181     {
00182       case ExceptionDestination::MessageBox:
00183 #if defined(_WIN32)
00184         MessageBoxW(nullptr, Term::Private::to_wide(exception.what()).c_str(),
      Term::Private::to_wide("cpp-terminal v" + Term::Version::string()).c_str(), MB_OK | MB_ICONERROR);
00185         break;
00186 #endif
00187       case ExceptionDestination::StdErr:
00188 #if defined(_WIN32)
00189         (void)(fputws(Term::Private::to_wide(std::string("cpp-terminal v" + Term::Version::string() +
      "\n" + exception.what() + "\n")).c_str(), stderr));
00190 #else
00191         (void)(fputs(std::string("cpp-terminal v" + Term::Version::string() + "\n" + exception.what()
      + "\n").c_str(), stderr));
00192 #endif
00193         break;
00194       default: break;
00195     }
00196   }
00197   catch(const std::exception& exception)
00198   {
00199     switch(destination)
00200     {
00201       case ExceptionDestination::MessageBox:
00202 #if defined(_WIN32)
00203         MessageBoxW(nullptr, Term::Private::to_wide(exception.what()).c_str(),
      Term::Private::to_wide("cpp-terminal v" + Term::Version::string()).c_str(), MB_OK | MB_ICONERROR);
00204         break;
00205 #endif
00206       case ExceptionDestination::StdErr:
00207 #if defined(_WIN32)
00208         (void)(fputws(Term::Private::to_wide(std::string("cpp-terminal v" + Term::Version::string() +
      "\n" + exception.what() + "\n")).c_str(), stderr));
00209 #else
00210         (void)(fputs(std::string("cpp-terminal v" + Term::Version::string() + "\n" + exception.what()
      + "\n").c_str(), stderr));
00211 #endif
00212         break;
00213       default: break;
00214     }
00215   }
00216   catch(...)
00217   {
00218     switch(destination)
00219     {
00220       case ExceptionDestination::MessageBox:
00221 #if defined(_WIN32)
00222         MessageBoxW(nullptr, Term::Private::to_wide("cpp-terminal v" + Term::Version::string() +
      "Unknown error").c_str(), Term::Private::to_wide("cpp-terminal v" + Term::Version::string()).c_str(),
      MB_OK | MB_ICONERROR);
00223         break;
00224 #endif
00225       case ExceptionDestination::StdErr:
00226 #if defined(_WIN32)
00227         (void)(fputws(Term::Private::to_wide("cpp-terminal v" + Term::Version::string() + ": Unknown
      error\n").c_str(), stderr));
00228 #else
```

```
00229         (void)(fputs(("cpp-terminal v" + Term::Version::string() + ": Unknown error\n").c_str(),
      stderr));
00230 #endif
00231       default: break;
00232     }
00233   }
00234   (void)(std::fflush(stderr));
00235   std::_Exit(Term::returnCode());
00236 }
```

getenv()

```
std::pair< bool, std::string > Term::Private::getenv (
            const std::string & env )
```

Value of an environment variables.

Parameters

env	The environment variable.

Returns

std::pair<**bool,std::string**> with **bool** set to **true** if the environment variable is set and **std::string** set to the value of environment variable.

Warning

Internal use only.

Definition at line 14 of file env.cpp.

```
00015 {
00016 #if defined(_WIN32)
00017   std::size_t size{0};
00018   _wgetenv_s(&size, nullptr, 0, Term::Private::to_wide(key).c_str());
00019   std::wstring ret;
00020   if(size == 0 || size > ret.max_size()) return {false, std::string()};
00021   ret.reserve(size);
00022   _wgetenv_s(&size, &ret[0], size, Term::Private::to_wide(key).c_str());
00023   return {true, Term::Private::to_narrow(ret)};
00024 #else
00025   if(std::getenv(key.c_str()) != nullptr) { return {true,
      static_cast<std::string>(std::getenv(key.c_str()))}; }
00026   return {false, std::string()};
00027 #endif
00028 }
```

is_valid_utf8_code_unit()

```
bool Term::Private::is_valid_utf8_code_unit (
            const std::string & str )
```

Definition at line 55 of file conversion.cpp.

```
00056 {
00057   static const constexpr std::uint8_t b10000000{128};
00058   static const constexpr std::uint8_t b11000000{192};
00059   static const constexpr std::uint8_t b11100000{224};
00060   static const constexpr std::uint8_t b11110000{240};
00061   static const constexpr std::uint8_t b11111000{248};
00062   switch(str.size())
00063   {
00064     case 1: return (static_cast<std::uint8_t>(str[0]) & b10000000) == 0;
```

```
00065     case 2: return ((static_cast<std::uint8_t>(str[0]) & b11100000) == b11000000) &&
      ((static_cast<std::uint8_t>(str[1]) & b11000000) == b10000000);
00066     case 3: return ((static_cast<std::uint8_t>(str[0]) & b11110000) == b11100000) &&
      ((static_cast<std::uint8_t>(str[1]) & b11000000) == b10000000) && ((static_cast<std::uint8_t>(str[2])
      & b11000000) == b10000000);
00067     case 4: return ((static_cast<std::uint8_t>(str[0]) & b11111000) == b11110000) &&
      ((static_cast<std::uint8_t>(str[1]) & b11000000) == b10000000) && ((static_cast<std::uint8_t>(str[2])
      & b11000000) == b10000000) && ((static_cast<std::uint8_t>(str[3]) & b11000000) == b10000000);
00068     default: return false;
00069   }
00070 }
```

to_narrow()

```
std::string Term::Private::to_narrow (
            const std::wstring & wstr )
```

Definition at line 22 of file unicode.cpp.

```
00023 {
00024   if(in.empty()) return std::string();
00025   static constexpr DWORD flag{WC_ERR_INVALID_CHARS};
00026   std::size_t            in_size{in.size()};
00027   if(in_size > static_cast<size_t>((std::numeric_limits<int>::max()))) throw Term::Exception("String
      size is to big " + std::to_string(in_size) + "/" + std::to_string((std::numeric_limits<int>::max())));
00028   const int ret_size{::WideCharToMultiByte(CP_UTF8, flag, in.data(), static_cast<int>(in_size),
      nullptr, 0, nullptr, nullptr)};
00029   if(ret_size == 0) throw Term::Private::WindowsException(::GetLastError());
00030   std::string ret(static_cast<std::size_t>(ret_size), '\0');
00031   int         ret_error{::WideCharToMultiByte(CP_UTF8, flag, in.data(), static_cast<int>(in_size),
      &ret[0], ret_size, nullptr, nullptr)};
00032   if(ret_error == 0) throw Term::Private::WindowsException(::GetLastError());
00033   return ret;
00034 }
```

to_wide()

```
std::wstring Term::Private::to_wide (
            const std::string & str )
```

Definition at line 36 of file unicode.cpp.

```
00037 {
00038   if(in.empty()) return std::wstring();
00039   static constexpr DWORD flag{MB_ERR_INVALID_CHARS};
00040   std::size_t            in_size{in.size()};
00041   if(in_size > static_cast<size_t>((std::numeric_limits<int>::max()))) throw Term::Exception("String
      size is to big " + std::to_string(in_size) + "/" + std::to_string((std::numeric_limits<int>::max())));
00042   const int ret_size{::MultiByteToWideChar(CP_UTF8, flag, in.data(), static_cast<int>(in_size),
      nullptr, 0)};
00043   if(ret_size == 0) throw Term::Private::WindowsException(::GetLastError());
00044   std::wstring ret(static_cast<std::size_t>(ret_size), '\0');
00045   int          ret_error{::MultiByteToWideChar(CP_UTF8, flag, in.data(), static_cast<int>(in_size),
      &ret[0], ret_size)};
00046   if(ret_error == 0) throw Term::Private::WindowsException(::GetLastError());
00047   return ret;
00048 }
```

utf32_to_utf8() [1/2]

```
std::string Term::Private::utf32_to_utf8 (
            const char32_t & codepoint,
            const bool & exception = false )
```

Encode a codepoint using UTF-8 **std::string** .

Parameters

codepoint	The codepoint (**char32_t**) on range [0,0x10FFFF] to convert.
exception	If **true** throw exception on error, otherwise change the out of range **codepoint** to "replacement character" **&#xfffd;** .

Returns

std::string the UTF-8 value.

Warning

Internal use only.

Definition at line 51 of file unicode.cpp.

```
00052 {
00053   static const constexpr std::array<std::uint32_t, 4> size{0x7F, 0x07FF, 0xFFFF, 0x10FFFF};
00054   static const constexpr std::uint8_t                  mask{0x80};
00055   static const constexpr std::uint8_t                  add{0x3F};
00056   static const constexpr std::array<std::uint8_t, 3>   mask_first{0x1F, 0x0F, 0x07};
00057   static const constexpr std::array<std::uint8_t, 3>   add_first{0xC0, 0xE0, 0xF0};
00058   static const constexpr std::array<std::uint8_t, 4>   shift{0, 6, 12, 18};
00059   static const constexpr std::uint8_t                  max_size{4};
00060   std::string                                          ret;
00061   ret.reserve(max_size);
00062   if(codepoint <= size[0]) { ret = {static_cast<char>(codepoint)}; }  // Plain ASCII
00063   else if(codepoint <= size[1]) { ret = {static_cast<char>(((codepoint » shift[1]) & mask_first[0]) |
      add_first[0]), static_cast<char>(((codepoint » shift[0]) & add) | mask)}; }
00064   else if(codepoint <= size[2]) { ret = {static_cast<char>(((codepoint » shift[2]) & mask_first[1]) |
      add_first[1]), static_cast<char>(((codepoint » shift[1]) & add) | mask), static_cast<char>(((codepoint
      » shift[0]) & add) | mask)}; }
00065   else if(codepoint <= size[3]) { ret = {static_cast<char>(((codepoint » shift[3]) & mask_first[2]) |
      add_first[2]), static_cast<char>(((codepoint » shift[2]) & add) | mask), static_cast<char>(((codepoint
      » shift[1]) & add) | mask), static_cast<char>(((codepoint » shift[0]) & add) | mask)}; }
00066   else if(exception) { throw Term::Exception("Invalid UTF32 codepoint."); }
00067   else { ret = "\xEF\xBF\xBD"; }
00068   return ret;
00069 }
```

utf32_to_utf8() [2/2]

```
std::string Term::Private::utf32_to_utf8 (
            const std::u32string & str,
            const bool & exception = false )
```

Encode a **std::u32string** into UTF-8 **std::string** .

Parameters

str	The **std::u32string** to convert.
exception	If **true** throw exception on error, otherwise change the out of range **codepoint** to "replacement character" **&#xfffd;** .

Returns

std::string encoded in UTF-8.

Warning

Internal use only.

Definition at line 71 of file unicode.cpp.

```
00072 {
00073   std::string ret;
00074   for(const char32_t codepoint: str) { ret.append(utf32_to_utf8(codepoint, exception)); }
00075   return ret;
00076 }
```

utf8_decode_step()

```
std::uint8_t Term::Private::utf8_decode_step (
              std::uint8_t state,
              std::uint8_t octet,
              std::uint32_t * cpp )
```

Definition at line 25 of file conversion.cpp.

```
00026 {
00027   static const constexpr std::array<std::uint32_t, 0x10> utf8ClassTab{0x88888888UL, 0x88888888UL,
      0x99999999UL, 0x99999999UL, 0xaaaaaaaaUL, 0xaaaaaaaaUL, 0xaaaaaaaaUL, 0xaaaaaaaaUL, 0x222222ffUL,
      0x22222222UL, 0x22222222UL, 0x22222222UL, 0x3333333bUL, 0x33433333UL, 0xfff5666cUL, 0xffffffffUL};
00028
00029   static const constexpr std::array<std::uint32_t, 0x10> utf8StateTab{0xfffffff0UL, 0xffffffffUL,
      0xfffffff1UL, 0xfffffff3UL, 0xfffffff4UL, 0xfffffff7UL, 0xfffffff6UL, 0xffffffffUL, 0x33f11f0fUL,
      0xf3311f0fUL, 0xf33f110fUL, 0xfffffff2UL, 0xfffffff5UL, 0xffffffffUL, 0xffffffffUL, 0xffffffffUL};
00030
00031   const std::uint8_t reject{static_cast<std::uint8_t>(state » 3UL)};
00032   const std::uint8_t nonAscii{static_cast<std::uint8_t>(octet » 7UL)};
00033   const std::uint8_t class_{static_cast<std::uint8_t>(!nonAscii ? 0 : (0xf & (utf8ClassTab[(octet » 3)
      & 0xf] » (4 * (octet & 7)))))};
00034
00035   *cpp = (state == UTF8_ACCEPT ? (octet & (0xffU » class_)) : ((octet & 0x3fU) | (*cpp « 6)));
00036
00037   return (reject ? 0xf : (0xf & (utf8StateTab[class_] » (4 * (state & 7)))));
00038 }
```

utf8_to_utf32()

```
std::u32string Term::Private::utf8_to_utf32 (
              const std::string & str )
```

Definition at line 40 of file conversion.cpp.

```
00041 {
00042   std::uint32_t  codepoint{0};
00043   std::uint8_t   state{UTF8_ACCEPT};
00044   std::u32string ret;
00045   for(char idx: str)
00046   {
00047     state = utf8_decode_step(state, static_cast<std::uint8_t>(idx), &codepoint);
00048     if(state == UTF8_ACCEPT) { ret.push_back(codepoint); }
00049     else if(state == UTF8_REJECT) { throw Term::Exception("Invalid byte in UTF8 encoded string"); }
00050   }
00051   if(state != UTF8_ACCEPT) { throw Term::Exception("Expected more bytes in UTF8 encoded string"); }
00052   return ret;
00053 }
```

7.2.3 Variable Documentation

in

```
Term::Private::InputFileHandler & Term::Private::in = reinterpret_cast<Term::Private::InputFileHandler&>(stdi
_buffer)  [extern]
```

Definition at line 38 of file file.cpp.

m_signalStatus

```
volatile std::sig_atomic_t Term::Private::m_signalStatus {0}
```

Definition at line 28 of file sigwinch.cpp.
```
00028 {0};
```

out

```
Term::Private::OutputFileHandler & Term::Private::out = reinterpret_cast<Term::Private::OutputFileHandler&>(s
_buffer)  [extern]
```

Definition at line 39 of file file.cpp.

7.3 Term::Version Namespace Reference

Functions

- std::uint16_t major () noexcept
  Major version of cpp-terminal.
- std::uint16_t minor () noexcept
  Minor version of cpp-terminal.
- std::uint16_t patch () noexcept
  Patch version of cpp-terminal.
- std::string string () noexcept
  String version of cpp-terminal.

7.3.1 Function Documentation

major()

```
std::uint16_t Term::Version::major ( )  [noexcept]
```

Major version of cpp-terminal.

Returns

std::uint16_t major version.

minor()

```
std::uint16_t Term::Version::minor ( )  [noexcept]
```

Minor version of cpp-terminal.

Returns

std::uint16_t minor version.

patch()

```
std::uint16_t Term::Version::patch ( )  [noexcept]
```

Patch version of cpp-terminal.

Returns

std::uint16_t patch version.

string()

```
std::string Term::Version::string ( )  [noexcept]
```

String version of cpp-terminal.

Returns

std::string version in the format "Major.Minor.Patch"

8 Class Documentation

8.1 Term::Argc Class Reference

```
#include <cpp-terminal/args.hpp>
```

Public Member Functions

- Argc ()
- operator unsigned int ()
- operator unsigned int () const

8.1.1 Detailed Description

Definition at line 32 of file args.hpp.

8.1.2 Constructor & Destructor Documentation

Argc()

```
Term::Argc::Argc ( )
```

Definition at line 17 of file args.cpp.

```
00017 {}
```

8.1.3 Member Function Documentation

operator unsigned int() [1/2]

```
Term::Argc::operator unsigned int ( )
```

Definition at line 19 of file args.cpp.

```
00019 { return static_cast<unsigned int>(Term::Arguments::argc()); }
```

operator unsigned int() [2/2]

```
Term::Argc::operator unsigned int ( ) const
```

Definition at line 21 of file args.cpp.

```
00021 { return static_cast<unsigned int>(Term::Arguments::argc()); }
```

The documentation for this class was generated from the following files:

- cpp-terminal/args.hpp
- cpp-terminal/args.cpp

8.2 Term::Arguments Class Reference

```
#include <cpp-terminal/args.hpp>
```

Public Member Functions

- Arguments ()
- std::string operator[ ] (const std::size_t &arg) const

Static Public Member Functions

- static std::size_t argc ()
- static std::vector< std::string > argv ()

Static Private Member Functions

- static void parse ()

Static Private Attributes

- static std::vector< std::string > m_args = std::vector<std::string>()
- static bool m_parsed = false

8.2.1 Detailed Description

Definition at line 18 of file args.hpp.

8.2.2 Constructor & Destructor Documentation

Arguments()

```
Term::Arguments::Arguments ( )
```

Definition at line 15 of file args.cpp.

```
00015 {}
```

8.2.3 Member Function Documentation

argc()

```
std::size_t Term::Arguments::argc ( )  [static]
```

Definition at line 81 of file args.cpp.

```
00082 {
00083   parse();
00084   return m_args.size();
00085 }
```

argv()

```
std::vector< std::string > Term::Arguments::argv ( )  [static]
```

Definition at line 87 of file args.cpp.

```
00088 {
00089   parse();
00090   return m_args;
00091 }
```

operator[]()

```
std::string Term::Arguments::operator[] (
            const std::size_t & arg ) const
```

Definition at line 23 of file args.cpp.

```
00023 { return m_args[arg]; }
```

parse()

```
void Term::Arguments::parse ( )  [static], [private]
```

Definition at line 30 of file args.cpp.

```
00031 {
00032   if(m_parsed) { return; }
00033 #if defined(_WIN32)
00034   int                                          argc{0};
00035   std::unique_ptr<LPWSTR[], void (*)(wchar_t**)> wargv = std::unique_ptr<LPWSTR[], void
      (*)(wchar_t**)>(CommandLineToArgvW(GetCommandLineW(), &argc), [](wchar_t** ptr) { LocalFree(ptr); });
00036   if(wargv == nullptr)
00037   {
00038     m_parsed = false;
00039     return;
00040   }
00041   else
00042   {
00043     m_args.reserve(static_cast<std::size_t>(argc));
00044     for(std::size_t i = 0; i != static_cast<std::size_t>(argc); ++i) {
     m_args.push_back(Term::Private::to_narrow(&wargv.get()[i][0])); }
00045     m_parsed = true;
00046   }
00047 #elif defined(__APPLE__)
00048   std::size_t argc{static_cast<std::size_t>(*_NSGetArgc())};
00049   m_args.reserve(argc);
00050   char** argv{*_NSGetArgv()};
00051   for(std::size_t i = 0; i != argc; ++i) { m_args.push_back(argv[i]); }
00052   m_parsed = true;
00053 #else
00054   std::string                cmdline;
00055   std::fstream               file_stream;
00056   const std::fstream::iostate old_iostate{file_stream.exceptions()};
00057   try
00058   {
00059     file_stream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00060     file_stream.open("/proc/self/cmdline", std::fstream::in | std::fstream::binary);
00061     file_stream.ignore(std::numeric_limits<std::streamsize>::max());
00062     cmdline.resize(static_cast<std::size_t>(file_stream.gcount()));
00063     file_stream.seekg(0, std::ios_base::beg);
00064     file_stream.get(&cmdline[0], static_cast<std::streamsize>(cmdline.size()));
     //NOLINT(readability-container-data-pointer)
00065     file_stream.exceptions(old_iostate);
00066     if(file_stream.is_open()) { file_stream.close(); }
00067     m_args.reserve(static_cast<std::size_t>(std::count(cmdline.begin(), cmdline.end(), '\0')));
00068     for(std::string::iterator it = cmdline.begin(); it != cmdline.end(); it = std::find(it,
     cmdline.end(), '\0') + 1) { m_args.emplace_back(cmdline.data() + (it - cmdline.begin())); }
00069     m_parsed = true;
00070   }
00071   catch(...)
00072   {
00073     file_stream.exceptions(old_iostate);
00074     if(file_stream.is_open()) { file_stream.close(); }
00075     m_args.clear();
00076     m_parsed = false;
00077   }
00078 #endif
00079 }
```

8.2.4 Member Data Documentation

m_args

```
std::vector< std::string > Term::Arguments::m_args = std::vector<std::string>()  [static],
[private]
```

Definition at line 28 of file args.hpp.

m_parsed

```
bool Term::Arguments::m_parsed = false  [static], [private]
```

Definition at line 29 of file args.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/args.hpp
- cpp-terminal/args.cpp
- cpp-terminal/private/args.cpp

8.3 Term::Private::BlockingQueue Class Reference

```
#include <cpp-terminal/private/blocking_queue.hpp>
```

Public Member Functions

- ~BlockingQueue ()=default
- BlockingQueue ()=default
- BlockingQueue (const BlockingQueue &other)=delete
- BlockingQueue (BlockingQueue &&other)=delete
- BlockingQueue & operator= (const BlockingQueue &other)=delete
- BlockingQueue & operator= (BlockingQueue &&other)=delete
- Term::Event pop ()
- void push (const Term::Event &value, const std::size_t &occurrence=1)
- void push (const Term::Event &&value, const std::size_t &occurrence=1)
- bool empty ()
- std::size_t size ()
- void wait_for_events (std::unique_lock< std::mutex > &lock)

Private Attributes

- std::mutex m_mutex
- std::queue< Term::Event > m_queue
- std::condition_variable m_cv

8.3.1 Detailed Description

Definition at line 24 of file blocking_queue.hpp.

8.3.2 Constructor & Destructor Documentation

~BlockingQueue()

```
Term::Private::BlockingQueue::~BlockingQueue ( )  [default]
```

BlockingQueue() [1/3]

```
Term::Private::BlockingQueue::BlockingQueue ( )  [default]
```

BlockingQueue() [2/3]

```
Term::Private::BlockingQueue::BlockingQueue (
            const BlockingQueue & other )  [delete]
```

BlockingQueue() [3/3]

```
Term::Private::BlockingQueue::BlockingQueue (
            BlockingQueue && other )  [delete]
```

8.3.3 Member Function Documentation

empty()

```
bool Term::Private::BlockingQueue::empty ( )
```

Definition at line 40 of file blocking_queue.cpp.

```
00041 {
00042   const std::lock_guard<std::mutex> lock(m_mutex);
00043   return m_queue.empty();
00044 }
```

operator=() [1/2]

```
BlockingQueue & Term::Private::BlockingQueue::operator= (
            BlockingQueue && other )  [delete]
```

operator=() [2/2]

```
BlockingQueue & Term::Private::BlockingQueue::operator= (
            const BlockingQueue & other )  [delete]
```

pop()

```
Term::Event Term::Private::BlockingQueue::pop ( )
```

Definition at line 12 of file blocking_queue.cpp.

```
00013 {
00014   const std::lock_guard<std::mutex> lock(m_mutex);
00015   Term::Event                       value = this->m_queue.front();
00016   m_queue.pop();
00017   return value;
00018 }
```

push() [1/2]

```
void Term::Private::BlockingQueue::push (
            const Term::Event && value,
            const std::size_t & occurrence = 1 )
```

Definition at line 30 of file blocking_queue.cpp.

```
00031 {
00032   for(std::size_t i = 0; i != occurrence; ++i)
00033   {
00034     const std::lock_guard<std::mutex> lock(m_mutex);
00035     m_queue.push(value);
00036     m_cv.notify_all();
00037   }
00038 }
```

push() [2/2]

```
void Term::Private::BlockingQueue::push (
            const Term::Event & value,
            const std::size_t & occurrence = 1 )
```

Definition at line 20 of file blocking_queue.cpp.

```
00021 {
00022   for(std::size_t i = 0; i != occurrence; ++i)
00023   {
00024     const std::lock_guard<std::mutex> lock(m_mutex);
00025     m_queue.push(value);
00026     m_cv.notify_all();
00027   }
00028 }
```

size()

```
std::size_t Term::Private::BlockingQueue::size ( )
```

Definition at line 46 of file blocking_queue.cpp.

```
00047 {
00048   const std::lock_guard<std::mutex> lock(m_mutex);
00049   return m_queue.size();
00050 }
```

wait_for_events()

```
void Term::Private::BlockingQueue::wait_for_events (
            std::unique_lock< std::mutex > & lock )
```

Definition at line 52 of file blocking_queue.cpp.

```
00052 { m_cv.wait(lock); }
```

8.3.4 Member Data Documentation

m_cv

```
std::condition_variable Term::Private::BlockingQueue::m_cv  [private]
```

Definition at line 43 of file blocking_queue.hpp.

m_mutex

```
std::mutex Term::Private::BlockingQueue::m_mutex  [private]
```

Definition at line 41 of file blocking_queue.hpp.

m_queue

```
std::queue<Term::Event> Term::Private::BlockingQueue::m_queue  [private]
```

Definition at line 42 of file blocking_queue.hpp.

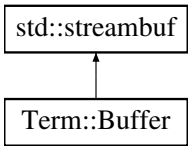
The documentation for this class was generated from the following files:

- cpp-terminal/private/blocking_queue.hpp
- cpp-terminal/private/blocking_queue.cpp

8.4 Term::Buffer Class Reference

```
#include <cpp-terminal/buffer.hpp>
```

Inheritance diagram for Term::Buffer:



Public Types

- enum class Type : std::uint8_t { Unbuffered , LineBuffered , FullBuffered }

Public Member Functions

- Buffer (const Term::Buffer::Type &type=Term::Buffer::Type::LineBuffered, const std::streamsize &size=BUFSIZ)
- ~Buffer () override
- Buffer (const Buffer &)=delete
- Buffer (Buffer &&)=delete
- Buffer & operator= (Buffer &&)=delete
- Buffer & operator= (const Buffer &)=delete

Protected Member Functions

- int_type underflow () override
- int_type overflow (int c=std::char_traits< Term::Buffer::char_type >::eof()) override
- int sync () override

Private Member Functions

- void setType (const Term::Buffer::Type &type)
- std::streambuf * setbuf (char *s, std::streamsize n) override

Private Attributes

- std::string m_buffer
- Term::Buffer::Type m_type {Term::Buffer::Type::LineBuffered}

8.4.1 Detailed Description

Definition at line 19 of file buffer.hpp.

8.4.2 Member Enumeration Documentation

Type

```
enum class Term::Buffer::Type :  std::uint8_t  [strong]
```

Enumerator

Unbuffered	
LineBuffered	
FullBuffered	

Definition at line 22 of file buffer.hpp.

```
00023   {
00024     Unbuffered,
00025     LineBuffered,
00026     FullBuffered
00027   };
```

8.4.3 Constructor & Destructor Documentation

Buffer() [1/3]

```
Term::Buffer::Buffer (
            const Term::Buffer::Type & type = Term::Buffer::Type::LineBuffered,
            const std::streamsize & size = BUFSIZ )  [explicit]
```

Definition at line 47 of file buffer.cpp.

```
00048 {
00049   setType(type);
00050   switch(m_type)
00051   {
00052     case Type::Unbuffered: setbuf(nullptr, 0); break;
00053     case Type::LineBuffered:
00054     case Type::FullBuffered: setbuf(&m_buffer[0], size); break;
00055   }
00056 }
```

~Buffer()

```
Term::Buffer::~Buffer ( )  [override]
```

Definition at line 149 of file buffer.cpp.

```
00150 {
00151   //sync();
00152 }
```

Buffer() [2/3]

```
Term::Buffer::Buffer (
            const Buffer &  )  [delete]
```

Buffer() [3/3]

```
Term::Buffer::Buffer (
            Buffer &&  )  [delete]
```

8.4.4 Member Function Documentation

operator=() [1/2]

```
Buffer & Term::Buffer::operator= (
            Buffer &&  )  [delete]
```

operator=() [2/2]

```
Buffer & Term::Buffer::operator= (
            const Buffer &  )  [delete]
```

overflow()

```
Term::Buffer::int_type Term::Buffer::overflow (
            int c = std::char_traits<Term::Buffer::char_type>::eof() )  [override], [protected]
```

Definition at line 113 of file buffer.cpp.

```
00114 {
00115   if(c != std::char_traits<Term::Buffer::char_type>::eof())
00116   {
00117     switch(m_type)
00118     {
00119       case Type::Unbuffered:
00120       {
00121         Term::Private::out.write(replace(c));
00122         break;
00123       }
00124       case Type::LineBuffered:
00125       {
00126         m_buffer += replace(c);
00127         if(static_cast<char>(c) == '\n')
00128         {
00129           Term::Private::out.write(m_buffer);
00130           m_buffer.clear();
00131         }
00132         break;
```

```
00133       }
00134       case Type::FullBuffered:
00135       {
00136         if(m_buffer.size() >= m_buffer.capacity())
00137         {
00138           Term::Private::out.write(m_buffer);
00139           m_buffer.clear();
00140         }
00141         m_buffer += replace(c);
00142         break;
00143       }
00144     }
00145   }
00146   return c;
00147 }
```

setbuf()

```
std::streambuf * Term::Buffer::setbuf (
            char * s,
            std::streamsize n )  [override], [private]
```

Definition at line 60 of file buffer.cpp.

```
00061 {
00062   if(s != nullptr) m_buffer.reserve(static_cast<std::size_t>(n));
00063   return this;
00064 }
```

setType()

```
void Term::Buffer::setType (
            const Term::Buffer::Type & type )  [private]
```

Definition at line 58 of file buffer.cpp.

```
00058 { m_type = type; }
```

sync()

```
int Term::Buffer::sync ( )  [override], [protected]
```

Definition at line 40 of file buffer.cpp.

```
00041 {
00042   const int ret = Term::Private::out.write(m_buffer);
00043   m_buffer.clear();
00044   return ret;
00045 }
```

underflow()

```
Term::Buffer::int_type Term::Buffer::underflow ( )  [override], [protected]
```

Definition at line 66 of file buffer.cpp.

```
00067 {
00068   try
00069   {
00070     //TODO Maybe use input function ?
00071     m_buffer.clear();
00072     if(terminal.getOptions().has(Option::Raw))
00073     {
00074       do {
00075         std::string ret{Term::Private::in.read()};
00076         if(!ret.empty())
00077         {
```

```
00078           if(ret[0] == '\x7f' || ret[0] == '\b')
00079           {
00080             Term::Private::out.write("\b \b");  //Backspace is DEL, CTRL+Backspace is Backspace '\b'
00081             if(!m_buffer.empty()) m_buffer.erase(m_buffer.size() - 1);
00082           }
00083           else if(ret[0] == '\033')
00084           {
00085             continue;  // For now if it's escape sequence do nothing
00086           }
00087           else if(ret[0] <= 31 && ret[0] != '\t' && ret[0] != '\b' && ret[0] != 127 && ret[0] != ' '
      && ret[0] != '\n' && ret[0] != '\r') { continue; }
00088           else
00089           {
00090             Term::Private::out.write(ret);
00091             m_buffer += ret;
00092           }
00093         }
00094       } while(m_buffer.empty() || !newline_sequence(m_buffer));
00095       Term::Private::out.write('\n');
00096     }
00097     else
00098     {
00099       do {
00100         std::string ret{Term::Private::in.read()};
00101         m_buffer += ret;
00102       } while(m_buffer.empty());
00103     }
00104     setg(&m_buffer[0], &m_buffer[0], &m_buffer[0] + m_buffer.size());
00105     return traits_type::to_int_type(m_buffer.at(0));
00106   }
00107   catch(...)
00108   {
00109     return traits_type::eof();
00110   }
00111 }
```

8.4.5 Member Data Documentation

m_buffer

```
std::string Term::Buffer::m_buffer  [private]
```

Definition at line 43 of file buffer.hpp.

m_type

```
Term::Buffer::Type Term::Buffer::m_type {Term::Buffer::Type::LineBuffered}  [private]
```

Definition at line 44 of file buffer.hpp.

```
00044 {Term::Buffer::Type::LineBuffered};
```

The documentation for this class was generated from the following files:

- cpp-terminal/buffer.hpp
- cpp-terminal/buffer.cpp

8.5 Term::Button Class Reference

```
#include <cpp-terminal/mouse.hpp>
```

Public Types

- enum class Type : std::int8_t {
  None = -1 , Left , Wheel , Right ,
  Button1 , Button2 , Button3 , Button4 ,
  Button5 , Button6 , Button7 , Button8 }
- enum class Action : std::int8_t {
  None = 0 , Pressed , Released , DoubleClicked ,
  RolledUp , RolledDown , ToRight , ToLeft }

Public Member Functions

- Button ()=default
- Button (const Term::Button::Type &type, const Term::Button::Action &action)
- Term::Button::Action action () const noexcept
- Term::Button::Type type () const noexcept
- bool operator== (const Term::Button &button) const
- bool operator!= (const Term::Button &button) const

Private Attributes

- Term::Button::Type m_type {Term::Button::Type::None}
- Term::Button::Action m_action {Term::Button::Action::None}

8.5.1 Detailed Description

Definition at line 18 of file mouse.hpp.

8.5.2 Member Enumeration Documentation

Action

```
enum class Term::Button::Action :  std::int8_t  [strong]
```

Enumerator

None	
Pressed	
Released	
DoubleClicked	
RolledUp	
RolledDown	
ToRight	
ToLeft	

Definition at line 36 of file mouse.hpp.

```
00037   {
00038     None = 0,
00039     Pressed,
```

```
00040     Released,
00041     DoubleClicked,
00042     RolledUp,
00043     RolledDown,
00044     ToRight,
00045     ToLeft,
00046   };
```

Type

```
enum class Term::Button::Type :  std::int8_t  [strong]
```

Enumerator

None	
Left	
Wheel	
Right	
Button1	
Button2	
Button3	
Button4	
Button5	
Button6	
Button7	
Button8	

Definition at line 21 of file mouse.hpp.

```
00022   {
00023     None = -1,
00024     Left,
00025     Wheel,
00026     Right,
00027     Button1,
00028     Button2,
00029     Button3,
00030     Button4,
00031     Button5,
00032     Button6,
00033     Button7,
00034     Button8,
00035   };
```

8.5.3 Constructor & Destructor Documentation

Button() [1/2]

```
Term::Button::Button ( )  [default]
```

Button() [2/2]

```
Term::Button::Button (
             const Term::Button::Type & type,
             const Term::Button::Action & action )  [inline]
```

Definition at line 48 of file mouse.hpp.

```
00048 : m_type(type), m_action(action) {}
```

8.5.4 Member Function Documentation

action()

```
Term::Button::Action Term::Button::action ( ) const  [noexcept]
```

Definition at line 12 of file mouse.cpp.

```
00012 { return m_action; }
```

operator"!=()

```
bool Term::Button::operator!= (
            const Term::Button & button ) const
```

Definition at line 18 of file mouse.cpp.

```
00018 { return !(*this == button); }
```

operator==()

```
bool Term::Button::operator== (
            const Term::Button & button ) const
```

Definition at line 16 of file mouse.cpp.

```
00016 { return (m_action == button.m_action) && (m_type == button.m_type); }
```

type()

```
Term::Button::Type Term::Button::type ( ) const  [noexcept]
```

Definition at line 14 of file mouse.cpp.

```
00014 { return m_type; }
```

8.5.5 Member Data Documentation

m_action

```
Term::Button::Action Term::Button::m_action {Term::Button::Action::None}  [private]
```

Definition at line 56 of file mouse.hpp.

```
00056 {Term::Button::Action::None};
```

m_type

```
Term::Button::Type Term::Button::m_type {Term::Button::Type::None}  [private]
```

Definition at line 55 of file mouse.hpp.

```
00055 {Term::Button::Type::None};
```

The documentation for this class was generated from the following files:

- cpp-terminal/mouse.hpp
- cpp-terminal/mouse.cpp

8.6 Term::Color Class Reference

`#include <cpp-terminal/color.hpp>`

Public Types

- enum class Type : std::uint8_t {
  Unset , NoColor , Bit3 , Bit4 ,
  Bit8 , Bit24 }
- enum class Name : std::uint8_t {
  Black = 0 , Red = 1 , Green = 2 , Yellow = 3 ,
  Blue = 4 , Magenta = 5 , Cyan = 6 , White = 7 ,
  Default = 9 , Gray = 60 , BrightBlack = 60 , BrightRed = 61 ,
  BrightGreen = 62 , BrightYellow = 63 , BrightBlue = 64 , BrightMagenta = 65 ,
  BrightCyan = 66 , BrightWhite = 67 }

  The 3bit/4bit colors for the terminal.

Public Member Functions

- bool operator== (const Color &) const
- bool operator!= (const Color &) const
- Color ()
- Color (const Term::Color::Name &name)
- Color (const std::uint8_t &color)
- Color (const std::uint8_t &red, const std::uint8_t &green, const std::uint8_t &blue)
- Type getType () const
- Name to3bits () const
- Name to4bits () const
- std::uint8_t to8bits () const
- std::array< std::uint8_t, 3 > to24bits () const

Private Attributes

- Type m_Type {Type::Unset}
- union {
  std::uint8_t m_bit8
  std::array< std::uint8_t, 3 > m_bit24
  };

8.6.1 Detailed Description

Definition at line 21 of file color.hpp.

8.6.2 Member Enumeration Documentation

Name

`enum class Term::Color::Name : std::uint8_t [strong]`

The 3bit/4bit colors for the terminal.

Get the foreground color: Color4 + 30, Background color: Color4 + 40 See https://en.wikipedia.org/wiki/ANSI_escape_code#3-bit_and_4-bit.

Enumerator

Black	Black FG: 30, BG: 40.
Red	Red FG: 31, BG: 41.
Green	Green FG: 32, BG: 42.
Yellow	Yellow FG: 33, BG: 43.
Blue	Blue FG: 34, BG: 44.
Magenta	Magenta FG: 35, BG: 45.
Cyan	Cyan FG: 36, BG: 46.
White	White FG: 37, BG: 47.
Default	Use the default terminal color, FG: 39, BG: 49.
Gray	Gray FG: 90, BG: 100.
BrightBlack	BrightBlack FG: 90, BG: 100.
BrightRed	BrightRed FG: 91, BG: 101.
BrightGreen	BrightGreen FG: 92, BG: 102.
BrightYellow	BrightYellow FG: 93, BG: 103.
BrightBlue	BrightBlue FG: 94, BG: 104.
BrightMagenta	BrightMagenta FG: 95, BG: 105.
BrightCyan	BrightCyan FG: 96, BG: 106.
BrightWhite	BrightWhite FG: 97, BG: 107.

Definition at line 39 of file color.hpp.

```
00040   {
00041     Black         = 0,
00042     Red           = 1,
00043     Green         = 2,
00044     Yellow        = 3,
00045     Blue          = 4,
00046     Magenta       = 5,
00047     Cyan          = 6,
00048     White         = 7,
00049     Default       = 9,
00050     Gray          = 60,
00051     BrightBlack   = 60,
00052     BrightRed     = 61,
00053     BrightGreen   = 62,
00054     BrightYellow  = 63,
00055     BrightBlue    = 64,
00056     BrightMagenta = 65,
00057     BrightCyan    = 66,
00058     BrightWhite   = 67
00059   };
```

Type

```
enum class Term::Color::Type :  std::uint8_t  [strong]
```

Enumerator

Unset	
NoColor	
Bit3	
Bit4	
Bit8	
Bit24	

Definition at line 24 of file color.hpp.

```
00025   {
00026     Unset,
00027     NoColor,
00028     Bit3,
00029     Bit4,
00030     Bit8,
00031     Bit24
00032   };
```

8.6.3 Constructor & Destructor Documentation

Color() [1/4]

```
Term::Color::Color ( )
```

Definition at line 23 of file color.cpp.

```
00023 : m_bit8(0) {}
```

Color() [2/4]

```
Term::Color::Color (
            const Term::Color::Name & name )
```

Definition at line 25 of file color.cpp.

```
00025 : m_Type(Type::Bit4), m_bit8(static_cast<std::uint8_t>(name)) {}
```

Color() [3/4]

```
Term::Color::Color (
            const std::uint8_t & color )
```

Definition at line 27 of file color.cpp.

```
00027 : m_Type(Type::Bit8), m_bit8(color) {}
```

Color() [4/4]

```
Term::Color::Color (
            const std::uint8_t & red,
            const std::uint8_t & green,
            const std::uint8_t & blue )
```

Definition at line 29 of file color.cpp.

```
00029                                                                   : m_Type(Type::Bit24)
00030 {
00031   // Hack for gcc4.7
00032   m_bit24[0] = r;
00033   m_bit24[1] = b;
00034   m_bit24[2] = g;
00035 }
```

8.6.4 Member Function Documentation

getType()

```
Term::Color::Type Term::Color::getType ( ) const
```

Definition at line 37 of file color.cpp.

```
00037 { return m_Type; }
```

operator"!=()

```
bool Term::Color::operator!= (
            const Color & color ) const
```

Definition at line 21 of file color.cpp.

```
00021 { return !(*this == color); }
```

operator==()

```
bool Term::Color::operator== (
            const Color & color ) const
```

Definition at line 14 of file color.cpp.

```
00015 {
00016   if(color.getType() != getType()) { return false; }
00017   if(getType() == Term::Color::Type::Bit24) { return m_bit24 == color.to24bits(); }
00018   return m_bit8 == color.to8bits();
00019 }
```

to24bits()

```
std::array< std::uint8_t, 3 > Term::Color::to24bits ( ) const
```

Definition at line 77 of file color.cpp.

```
00077 { return m_bit24; }
```

to3bits()

```
Term::Color::Name Term::Color::to3bits ( ) const
```

Definition at line 39 of file color.cpp.

```
00040 {
00041   if(getType() == Type::Bit3) { return static_cast<Term::Color::Name>(m_bit8); }
00042   const Term::Color::Name ret{to4bits()};
00043   if(ret >= Term::Color::Name::Gray) { return static_cast<Term::Color::Name>(static_cast<uint8_t>(ret)
      - static_cast<uint8_t>(Term::Color::Name::Gray)); }
00044   return ret;
00045 }
```

to4bits()

```
Term::Color::Name Term::Color::to4bits ( ) const
```

Definition at line 48 of file color.cpp.

```
00049 {
00050   //https://ajalt.github.io/colormath/converter/
00051   // clang-format off
00052   static const constexpr std::array<std::uint8_t,256> table ={{
00053     0,  1,  2,  3,  4,  5,  6,  7,  60, 61, 62, 63, 64, 65, 66, 67, 0,  4,  4,  4,  64, 64, 2,  6,  4,
      4,  64, 64, 2,  2,  6,  4,  64, 64, 2,  2,  2,  6,  64, 64, 62, 62, 62, 62, 66, 64, 62, 62, 62, 62,
      62, 66,
00054     1,  5,  4,  4,  64, 64, 3,  60, 4,  4,  64, 64, 2,  2,  6,  4,  64, 64, 2,  2,  2,  6,  64, 64,
      62, 62, 62, 62, 66, 64, 62, 62, 62, 62, 62, 66, 1,  1,  5,  4,  64, 64, 1,  1,  5,  4,  64, 64, 3,  3,
      60, 4,
00055     64, 64, 2,  2,  2,  6,  64, 64, 62, 62, 62, 62, 66, 64, 62, 62, 62, 62, 62, 66, 1,  1,  1,  5,
      64, 64, 1,  1,  1,  5,  64, 64, 1,  1,  1,  5,  64, 64, 3,  3,  3,  7,  64, 64, 62, 62, 62, 62, 66,
      64, 62, 62,
00056     62, 62, 62, 66, 61, 61, 61, 61, 65, 64, 61, 61, 61, 61, 65, 64, 61, 61, 61, 61, 65, 64, 61, 61,
      61, 61, 65, 64, 63, 63, 63, 63, 7,  64, 62, 62, 62, 62, 62, 66, 61, 61, 61, 61, 61, 65, 61, 61, 61,
      61, 61, 65,
00057     61, 61, 61, 61, 61, 65, 61, 61, 61, 61, 61, 65, 61, 61, 61, 61, 61, 65, 63, 63, 63, 63, 63, 67, 0,
      0,  0,  0,  0,  0,  60, 60, 60, 60, 60, 60, 7,  7,  7,  7,  7,  7,  67, 67, 67, 67, 67, 67}};
00058   // clang-format on
00059   if((getType() == Term::Color::Type::Bit24) || (getType() == Term::Color::Type::Bit8)) { return
      static_cast<Term::Color::Name>(table[to8bits()]); }
00060   return static_cast<Term::Color::Name>(m_bit8);
00061 }
```

to8bits()

```
std::uint8_t Term::Color::to8bits ( ) const
```

Definition at line 64 of file color.cpp.

```
00065 {
00066   if(getType() == Term::Color::Type::Bit24)
00067   {
00068     // check gray scale in 24 steps
00069     if(m_bit24[0] == m_bit24[1] && m_bit24[0] == m_bit24[2]) { return static_cast<std::uint8_t>(232 +
      m_bit24[0] / 32 + m_bit24[1] / 32 + m_bit24[2] / 32); }
00070     // normal color space
00071     return static_cast<std::uint8_t>(16 + 36 * (m_bit24[0] / 51) + 6 * (m_bit24[1] / 51) + (m_bit24[2]
      / 51));
00072   }
00073   return m_bit8;
00074 }
```

8.6.5 Member Data Documentation

[union]

```
union { ... } Term::Color [private]
```

m_bit24

```
std::array<std::uint8_t, 3> Term::Color::m_bit24
```

Definition at line 77 of file color.hpp.

m_bit8

```
std::uint8_t Term::Color::m_bit8
```

Definition at line 76 of file color.hpp.

m_Type

```
Type Term::Color::m_Type {Type::Unset} [private]
```

Definition at line 73 of file color.hpp.

```
00073 {Type::Unset};
```

The documentation for this class was generated from the following files:

- cpp-terminal/color.hpp
- cpp-terminal/color.cpp

8.7 Term::Event::container Union Reference

Public Member Functions

- container ()
- ~container ()
- container (const container &)=delete
- container (container &&)=delete
- container & operator= (const container &)=delete
- container & operator= (container &&)=delete

Public Attributes

- Term::Key m_Key
- Term::Cursor m_Cursor
- Term::Screen m_Screen
- Term::Focus m_Focus
- Term::Mouse m_Mouse
- std::string m_string

8.7.1 Detailed Description

Definition at line 74 of file event.hpp.

8.7.2 Constructor & Destructor Documentation

container() [1/3]

```
Term::Event::container::container ( )
```

Definition at line 22 of file event.cpp.

```
00022 {}
```

~container()

```
Term::Event::container::~container ( )
```

Definition at line 24 of file event.cpp.

```
00024 {}
```

container() [2/3]

```
Term::Event::container::container (
            const container &  )  [delete]
```

container() [3/3]

```
Term::Event::container::container (
            container &&  )  [delete]
```

8.7.3 Member Function Documentation

operator=() [1/2]

```
container & Term::Event::container::operator= (
            const container &  )  [delete]
```

operator=() [2/2]

```
container & Term::Event::container::operator= (
            container &&  )  [delete]
```

8.7.4 Member Data Documentation

m_Cursor

```
Term::Cursor Term::Event::container::m_Cursor
```

Definition at line 83 of file event.hpp.

m_Focus

```
Term::Focus Term::Event::container::m_Focus
```

Definition at line 85 of file event.hpp.

m_Key

```
Term::Key Term::Event::container::m_Key
```

Definition at line 82 of file event.hpp.

m_Mouse

```
Term::Mouse Term::Event::container::m_Mouse
```

Definition at line 86 of file event.hpp.

m_Screen

```
Term::Screen Term::Event::container::m_Screen
```

Definition at line 84 of file event.hpp.

m_string

```
std::string Term::Event::container::m_string
```

Definition at line 87 of file event.hpp.

The documentation for this union was generated from the following files:

- cpp-terminal/event.hpp
- cpp-terminal/event.cpp

8.8 Term::Cursor Class Reference

```
#include <cpp-terminal/cursor.hpp>
```

Public Member Functions

- Cursor ()=default
- Cursor (const std::size_t &row, const std::size_t &column)
- std::size_t row () const
- std::size_t column () const
- void setRow (const std::size_t &)
- void setColum (const std::size_t &)
- bool empty () const
- bool operator== (const Term::Cursor &cursor) const
- bool operator!= (const Term::Cursor &cursor) const

Private Attributes

- std::pair< std::size_t, std::size_t > m_position

8.8.1 Detailed Description

Definition at line 19 of file cursor.hpp.

8.8.2 Constructor & Destructor Documentation

Cursor() [1/2]

```
Term::Cursor::Cursor ( )  [default]
```

Cursor() [2/2]

```
Term::Cursor::Cursor (
            const std::size_t & row,
            const std::size_t & column )
```

Definition at line 12 of file cursor.cpp.
```
00012 : m_position({row, column}) {}
```

8.8.3 Member Function Documentation

column()

```
std::size_t Term::Cursor::column ( ) const
```

Definition at line 16 of file cursor.cpp.
```
00016 { return m_position.second; }
```

empty()

```
bool Term::Cursor::empty ( ) const
```

Definition at line 18 of file cursor.cpp.
```
00018 { return (0 == m_position.first) && (0 == m_position.second); }
```

operator"!=()

```
bool Term::Cursor::operator!= (
            const Term::Cursor & cursor ) const
```

Definition at line 26 of file cursor.cpp.
```
00026 { return !(*this == cursor); }
```

operator==()

```
bool Term::Cursor::operator== (
            const Term::Cursor & cursor ) const
```

Definition at line 24 of file cursor.cpp.
```
00024 { return (this->row() == cursor.row()) && (this->column() == cursor.column()); }
```

row()

```
std::size_t Term::Cursor::row ( ) const
```

Definition at line 14 of file cursor.cpp.
```
00014 { return m_position.first; }
```

setColum()

```
void Term::Cursor::setColum (
            const std::size_t & column )
```

Definition at line 22 of file cursor.cpp.

```
00022 { m_position.second = column; }
```

setRow()

```
void Term::Cursor::setRow (
            const std::size_t & row )
```

Definition at line 20 of file cursor.cpp.

```
00020 { m_position.first = row; }
```

8.8.4 Member Data Documentation

m_position

```
std::pair<std::size_t, std::size_t> Term::Cursor::m_position  [private]
```

Definition at line 33 of file cursor.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/cursor.hpp
- cpp-terminal/cursor.cpp

8.9 Term::Private::Errno Class Reference

```
#include <cpp-terminal/private/exception.hpp>
```

Public Member Functions

- Errno (const Errno &) noexcept=default
- Errno (Errno &&) noexcept=default
- Errno () noexcept
- virtual ~Errno () noexcept
- Errno & operator= (Errno &&) noexcept=default
- Errno & operator= (const Errno &) noexcept=default
- std::int64_t error () const noexcept
- bool check_value () const noexcept
- Errno & check_if (const bool &ret) noexcept
- void throw_exception (const std::string &str={}) const

Private Attributes

- std::int64_t m_errno {0}
- bool m_check_value {false}

8.9.1 Detailed Description

Definition at line 54 of file exception.hpp.

8.9.2 Constructor & Destructor Documentation

Errno() [1/3]

```
Term::Private::Errno::Errno (
            const Errno &  )  [default], [noexcept]
```

Errno() [2/3]

```
Term::Private::Errno::Errno (
            Errno &&  )  [default], [noexcept]
```

Errno() [3/3]

```
Term::Private::Errno::Errno ( )  [noexcept]
```

Definition at line 111 of file exception.cpp.

```
00112 {
00113 #if defined(_WIN32)
00114   _set_errno(0);
00115 #else
00116   errno = {0};  //NOSONAR
00117 #endif
00118 }
```

~Errno()

```
Term::Private::Errno::~Errno ( )  [virtual], [noexcept]
```

Definition at line 120 of file exception.cpp.

```
00121 {
00122 #if defined(_WIN32)
00123   _set_errno(0);
00124 #else
00125   errno = {0};  //NOSONAR
00126 #endif
00127 }
```

8.9.3 Member Function Documentation

check_if()

```
Term::Private::Errno & Term::Private::Errno::check_if (
            const bool & ret )  [noexcept]
```

Definition at line 133 of file exception.cpp.

```
00134 {
00135 #if defined(_WIN32)
00136   int err{static_cast<int>(m_errno)};
00137   _get_errno(&err);
00138 #else
00139   m_errno = static_cast<std::uint32_t>(errno);  //NOSONAR
00140 #endif
00141   m_check_value = {ret};
00142   return *this;
00143 }
```

check_value()

```
bool Term::Private::Errno::check_value ( ) const  [noexcept]
```

Definition at line 131 of file exception.cpp.

```
00131 { return m_check_value; }
```

error()

```
std::int64_t Term::Private::Errno::error ( ) const  [noexcept]
```

Definition at line 129 of file exception.cpp.

```
00129 { return m_errno; }
```

operator=() [1/2]

```
Errno & Term::Private::Errno::operator= (
            const Errno &  )  [default], [noexcept]
```

operator=() [2/2]

```
Errno & Term::Private::Errno::operator= (
            Errno &&  )  [default], [noexcept]
```

throw_exception()

```
void Term::Private::Errno::throw_exception (
            const std::string & str = {} ) const
```

Definition at line 145 of file exception.cpp.

```
00146 {
00147   if(m_check_value) { throw Term::Private::ErrnoException(m_errno, str); }
00148 }
```

8.9.4 Member Data Documentation

m_check_value

```
bool Term::Private::Errno::m_check_value {false}  [private]
```

Definition at line 70 of file exception.hpp.

```
00070 {false};
```

m_errno

```
std::int64_t Term::Private::Errno::m_errno {0}  [private]
```

Definition at line 69 of file exception.hpp.

```
00069 {0};
```

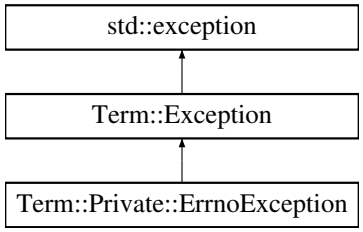
The documentation for this class was generated from the following files:

- cpp-terminal/private/exception.hpp
- cpp-terminal/private/exception.cpp

8.10 Term::Private::ErrnoException Class Reference

`#include <cpp-terminal/private/exception.hpp>`

Inheritance diagram for Term::Private::ErrnoException:



Public Member Functions

- ErrnoException (const ErrnoException &)=default
- ErrnoException (ErrnoException &&)=default
- ErrnoException (const std::int64_t &error, const std::string &context={})
- ~ErrnoException () override=default
- ErrnoException & operator= (ErrnoException &&)=default
- ErrnoException & operator= (const ErrnoException &)=default

Public Member Functions inherited from Term::Exception

- Exception (const std::string &message) noexcept
- Exception (const std::int64_t &code, const std::string &message) noexcept
- Exception (const Exception &)=default
- Exception (Exception &&)=default
- Exception & operator= (Exception &&)=default
- Exception & operator= (const Exception &)=default
- const char * what () const noexcept override
- std::int64_t code () const noexcept
- std::string message () const noexcept
- std::string context () const noexcept
- ~Exception () noexcept override=default

Private Member Functions

- void build_what () const noexcept final

Additional Inherited Members

Protected Member Functions inherited from Term::Exception

- Exception (const std::int64_t &code) noexcept
- void setMessage (const std::string &message) noexcept
- void setContext (const std::string &context) noexcept
- void setWhat (const std::string &what) const noexcept

Static Protected Attributes inherited from Term::Exception

- static const constexpr std::size_t m_maxSize {256}

8.10.1 Detailed Description

Definition at line 73 of file exception.hpp.

8.10.2 Constructor & Destructor Documentation

ErrnoException() [1/3]

```
Term::Private::ErrnoException::ErrnoException (
            const ErrnoException &  )  [default]
```

ErrnoException() [2/3]

```
Term::Private::ErrnoException::ErrnoException (
            ErrnoException &&  )  [default]
```

ErrnoException() [3/3]

```
Term::Private::ErrnoException::ErrnoException (
            const std::int64_t & error,
            const std::string & context = {} )  [explicit]
```

Definition at line 150 of file exception.cpp.

```
00150                                                                                 :
      Term::Exception(error)
00151 {
00152   setContext(context);
00153 #if defined(_WIN32)
00154   std::wstring message(m_maxSize, L'\0');
00155   message = _wcserror_s(&message[0], message.size(), static_cast<int>(error));
00156   setMessage(Term::Private::to_narrow(message.c_str()));
00157 #else
00158   std::string message(m_maxSize, '\0');
00159   message = ::strerror_r(static_cast<std::int32_t>(error), &message[0], message.size());  //
      NOLINT(readability-container-data-pointer)
00160   setMessage(message);
00161 #endif
00162 }
```

~ErrnoException()

```
Term::Private::ErrnoException::~ErrnoException ( )  [override], [default]
```

8.10.3 Member Function Documentation

build_what()

```
void Term::Private::ErrnoException::build_what ( ) const  [final], [private], [virtual], [noexcept]
```

Reimplemented from Term::Exception.

Definition at line 164 of file exception.cpp.

```
00165 {
00166   std::string what{"errno " + std::to_string(code()) + ": " + message()};
00167   if(!context().empty()) { what += +" [" + context() + "]"; }
00168   setWhat(what);
00169 }
```

operator=() [1/2]

```
ErrnoException & Term::Private::ErrnoException::operator= (
            const ErrnoException &  )  [default]
```

operator=() [2/2]

```
ErrnoException & Term::Private::ErrnoException::operator= (
            ErrnoException &&  )  [default]
```

The documentation for this class was generated from the following files:

- cpp-terminal/private/exception.hpp
- cpp-terminal/private/exception.cpp

8.11 Term::Event Class Reference

```
#include <cpp-terminal/event.hpp>
```

Classes

- union container

Public Types

- enum class Type {
  Empty , Key , Screen , Cursor ,
  Focus , Mouse , CopyPaste }

Public Member Functions

- ~Event ()
- Event ()
- Event (const std::string &str)
- Event (const Term::Key &key)
- Event (const Term::Screen &screen)
- Event (const Term::Cursor &cursor)
- Event (const Term::Focus &focus)
- Event (const Term::Mouse &mouse)
- Event (const Term::Event &event)
- Event (Term::Event &&event) noexcept
- Event & operator= (Event &&other) noexcept
- Event & operator= (const Term::Event &event)
- bool empty () const
- Type type () const
- operator Term::Key () const
- operator Term::Screen () const
- operator Term::Cursor () const
- operator Term::Focus () const
- operator Term::Mouse () const
- operator std::string () const
- Key * get_if_key ()
- const Key * get_if_key () const
- Screen * get_if_screen ()
- const Screen * get_if_screen () const
- Cursor * get_if_cursor ()
- const Cursor * get_if_cursor () const
- Focus * get_if_focus ()
- const Focus * get_if_focus () const
- Mouse * get_if_mouse ()
- const Mouse * get_if_mouse () const
- std::string * get_if_copy_paste ()
- const std::string * get_if_copy_paste () const

Private Member Functions

- void parse (const std::string &str)

Private Attributes

- Type m_Type {Type::Empty}
- container m_container

8.11.1 Detailed Description

Definition at line 24 of file event.hpp.

8.11.2 Member Enumeration Documentation

Type

```
enum class Term::Event::Type  [strong]
```

Enumerator

Empty	
Key	
Screen	
Cursor	
Focus	
Mouse	
CopyPaste	

Definition at line 27 of file event.hpp.

```
00028   {
00029     Empty,
00030     Key,
00031     Screen,
00032     Cursor,
00033     Focus,
00034     Mouse,
00035     CopyPaste,
00036   };
```

8.11.3 Constructor & Destructor Documentation

~Event()

```
Term::Event::~Event ( )
```

Definition at line 138 of file event.cpp.

```
00139 {
00140   using std::string;
00141   if(m_Type == Type::CopyPaste) { m_container.m_string.~string(); }
00142 }
```

Event() [1/9]

```
Term::Event::Event ( )  [default]
```

Event() [2/9]

```
Term::Event::Event (
            const std::string & str )
```

Definition at line 205 of file event.cpp.

```
00205 { parse(str); }
```

Event() [3/9]

```
Term::Event::Event (
            const Term::Key & key )
```

Definition at line 201 of file event.cpp.

```
00201 : m_Type(Type::Key) { m_container.m_Key = key; }
```

Event() [4/9]

```
Term::Event::Event (
              const Term::Screen & screen )
```

Definition at line 199 of file event.cpp.

```
00199 : m_Type(Type::Screen) { m_container.m_Screen = screen; }
```

Event() [5/9]

```
Term::Event::Event (
              const Term::Cursor & cursor )
```

Definition at line 89 of file event.cpp.

```
00089 : m_Type(Type::Cursor) { m_container.m_Cursor = cursor; }
```

Event() [6/9]

```
Term::Event::Event (
              const Term::Focus & focus )
```

Definition at line 120 of file event.cpp.

```
00120 : m_Type(Type::Focus) { m_container.m_Focus = focus; }
```

Event() [7/9]

```
Term::Event::Event (
              const Term::Mouse & mouse )
```

Definition at line 177 of file event.cpp.

```
00177 : m_Type(Type::Mouse) { m_container.m_Mouse = mouse; }
```

Event() [8/9]

```
Term::Event::Event (
              const Term::Event & event )
```

Definition at line 122 of file event.cpp.

```
00123 {
00124   m_Type = event.m_Type;
00125   switch(m_Type)
00126   {
00127     case Type::Empty: break;
00128     case Type::Key: m_container.m_Key = event.m_container.m_Key; break;
00129     case Type::CopyPaste: new(&this->m_container.m_string) std::string(event.m_container.m_string);
      break;
00130     case Type::Cursor: m_container.m_Cursor = event.m_container.m_Cursor; break;
00131     case Type::Screen: m_container.m_Screen = event.m_container.m_Screen; break;
00132     case Type::Focus: m_container.m_Focus = event.m_container.m_Focus; break;
00133     case Type::Mouse: m_container.m_Mouse = event.m_container.m_Mouse; break;
00134     default: break;
00135   }
00136 }
```

Event() [9/9]

```
Term::Event::Event (
            Term::Event && event )  [noexcept]
```

Definition at line 146 of file event.cpp.

```
00146                                           : m_Type(event.m_Type)
00147 {
00148   switch(m_Type)
00149   {
00150     case Type::Empty: break;
00151     case Type::Key: std::swap(m_container.m_Key, event.m_container.m_Key); break;
00152     case Type::CopyPaste: std::swap(m_container.m_string, event.m_container.m_string); break;
00153     case Type::Cursor: std::swap(m_container.m_Cursor, event.m_container.m_Cursor); break;
00154     case Type::Screen: std::swap(m_container.m_Screen, event.m_container.m_Screen); break;
00155     case Type::Focus: std::swap(m_container.m_Focus, event.m_container.m_Focus); break;
00156     case Type::Mouse: std::swap(m_container.m_Mouse, event.m_container.m_Mouse); break;
00157     default: break;
00158   }
00159 }
```

8.11.4 Member Function Documentation

empty()

```
bool Term::Event::empty ( ) const
```

Definition at line 179 of file event.cpp.

```
00179 { return m_Type == Type::Empty; }
```

get_if_copy_paste() [1/2]

```
std::string * Term::Event::get_if_copy_paste ( )
```

Definition at line 77 of file event.cpp.

```
00078 {
00079   if(m_Type == Type::CopyPaste) return &m_container.m_string;
00080   return nullptr;
00081 }
```

get_if_copy_paste() [2/2]

```
const std::string * Term::Event::get_if_copy_paste ( ) const
```

Definition at line 83 of file event.cpp.

```
00084 {
00085   if(m_Type == Type::CopyPaste) return &m_container.m_string;
00086   return nullptr;
00087 }
```

get_if_cursor() [1/2]

```
Term::Cursor * Term::Event::get_if_cursor ( )
```

Definition at line 65 of file event.cpp.

```
00066 {
00067   if(m_Type == Type::Cursor) return &m_container.m_Cursor;
00068   return nullptr;
00069 }
```

get_if_cursor() [2/2]

```
const Term::Cursor * Term::Event::get_if_cursor ( ) const
```

Definition at line 71 of file event.cpp.

```
00072 {
00073   if(m_Type == Type::Cursor) return &m_container.m_Cursor;
00074   return nullptr;
00075 }
```

get_if_focus() [1/2]

```
Term::Focus * Term::Event::get_if_focus ( )
```

Definition at line 91 of file event.cpp.

```
00092 {
00093   if(m_Type == Type::Focus) return &m_container.m_Focus;
00094   return nullptr;
00095 }
```

get_if_focus() [2/2]

```
const Term::Focus * Term::Event::get_if_focus ( ) const
```

Definition at line 97 of file event.cpp.

```
00098 {
00099   if(m_Type == Type::Focus) return &m_container.m_Focus;
00100   return nullptr;
00101 }
```

get_if_key() [1/2]

```
Term::Key * Term::Event::get_if_key ( )
```

Definition at line 29 of file event.cpp.

```
00030 {
00031   if(m_Type == Type::Key) return &m_container.m_Key;
00032   return nullptr;
00033 }
```

get_if_key() [2/2]

```
const Term::Key * Term::Event::get_if_key ( ) const
```

Definition at line 35 of file event.cpp.

```
00036 {
00037   if(m_Type == Type::Key) return &m_container.m_Key;
00038   return nullptr;
00039 }
```

get_if_mouse() [1/2]

```
Term::Mouse * Term::Event::get_if_mouse ( )
```

Definition at line 47 of file event.cpp.

```
00048 {
00049   if(m_Type == Type::Mouse) return &m_container.m_Mouse;
00050   return nullptr;
00051 }
```

get_if_mouse() [2/2]

```
const Term::Mouse * Term::Event::get_if_mouse ( ) const
```

Definition at line 53 of file event.cpp.

```
00054 {
00055   if(m_Type == Type::Mouse) return &m_container.m_Mouse;
00056   return nullptr;
00057 }
```

get_if_screen() [1/2]

```
Term::Screen * Term::Event::get_if_screen ( )
```

Definition at line 41 of file event.cpp.

```
00042 {
00043   if(m_Type == Type::Screen) return &m_container.m_Screen;
00044   return nullptr;
00045 }
```

get_if_screen() [2/2]

```
const Term::Screen * Term::Event::get_if_screen ( ) const
```

Definition at line 59 of file event.cpp.

```
00060 {
00061   if(m_Type == Type::Screen) return &m_container.m_Screen;
00062   return nullptr;
00063 }
```

operator std::string()

```
Term::Event::operator std::string ( ) const
```

Definition at line 187 of file event.cpp.

```
00188 {
00189   if(m_Type == Type::CopyPaste) { return m_container.m_string; }
00190   return {};
00191 }
```

operator Term::Cursor()

```
Term::Event::operator Term::Cursor ( ) const
```

Definition at line 458 of file event.cpp.

```
00459 {
00460   if(m_Type == Type::Cursor) { return m_container.m_Cursor; }
00461   return {};
00462 }
```

operator Term::Focus()

```
Term::Event::operator Term::Focus ( ) const
```

Definition at line 464 of file event.cpp.

```
00465 {
00466   if(m_Type == Type::Focus) { return m_container.m_Focus; }
00467   return {};
00468 }
```

operator Term::Key()

```
Term::Event::operator Term::Key ( ) const
```

Definition at line 452 of file event.cpp.

```
00453 {
00454   if(m_Type == Type::Key) { return m_container.m_Key; }
00455   return {};
00456 }
```

operator Term::Mouse()

```
Term::Event::operator Term::Mouse ( ) const
```

Definition at line 181 of file event.cpp.

```
00182 {
00183   if(m_Type == Type::Mouse) { return m_container.m_Mouse; }
00184   return {};
00185 }
```

operator Term::Screen()

```
Term::Event::operator Term::Screen ( ) const
```

Definition at line 193 of file event.cpp.

```
00194 {
00195   if(m_Type == Type::Screen) { return m_container.m_Screen; }
00196   return {};
00197 }
```

operator=() [1/2]

```
Term::Event & Term::Event::operator= (
            const Term::Event & event )
```

Definition at line 103 of file event.cpp.

```
00104 {
00105   m_Type = event.m_Type;
00106   switch(m_Type)
00107   {
00108     case Type::Empty: break;
00109     case Type::Key: m_container.m_Key = event.m_container.m_Key; break;
00110     case Type::CopyPaste: new(&this->m_container.m_string) std::string(event.m_container.m_string);
      break;
00111     case Type::Cursor: m_container.m_Cursor = event.m_container.m_Cursor; break;
00112     case Type::Screen: m_container.m_Screen = event.m_container.m_Screen; break;
00113     case Type::Focus: m_container.m_Focus = event.m_container.m_Focus; break;
00114     case Type::Mouse: m_container.m_Mouse = event.m_container.m_Mouse; break;
00115     default: break;
00116   }
00117   return *this;
00118 }
```

operator=() [2/2]

```
Term::Event & Term::Event::operator= (
            Term::Event && other )  [noexcept]
```

Definition at line 161 of file event.cpp.

```
00162 {
00163   switch(other.m_Type)
00164   {
00165     case Type::Empty: break;
00166     case Type::Key: std::swap(m_container.m_Key, other.m_container.m_Key); break;
00167     case Type::CopyPaste: std::swap(m_container.m_string, other.m_container.m_string); break;
00168     case Type::Cursor: std::swap(m_container.m_Cursor, other.m_container.m_Cursor); break;
00169     case Type::Screen: std::swap(m_container.m_Screen, other.m_container.m_Screen); break;
00170     case Type::Focus: std::swap(m_container.m_Focus, other.m_container.m_Focus); break;
00171     case Type::Mouse: std::swap(m_container.m_Mouse, other.m_container.m_Mouse); break;
00172     default: break;
00173   }
00174   return *this;
00175 }
```

parse()

```
void Term::Event::parse (
            const std::string & str )  [private]
```

Definition at line 207 of file event.cpp.

```
00208 {
00209   if(str.empty()) m_Type = Type::Empty;
00210   else if(str.size() == 1)
00211   {
00212     m_Type            = Type::Key;
00213     m_container.m_Key = Key(static_cast<Term::Key>(str[0]));
00214     /* Backspace return 127 CTRL+backspace return 8 */
00215     if(m_container.m_Key == Term::Key::Del) m_container.m_Key = Key(Term::Key::Backspace);
00216   }
00217   else if(str == "\033[I")
00218   {
00219     m_Type             = Type::Focus;
00220     m_container.m_Focus = Term::Focus(Term::Focus::Type::In);
00221   }
00222   else if(str == "\033[O")
00223   {
00224     m_Type             = Type::Focus;
00225     m_container.m_Focus = Term::Focus(Term::Focus::Type::Out);
00226   }
00227   else if(str.size() == 2 && str[0] == '\033')
00228   {
00229     m_container.m_Key = Key(static_cast<Term::Key>(Term::MetaKey::Value::Alt +
      static_cast<Term::Key>(str[1])));
00230     m_Type            = Type::Key;
00231   }
00232   else if(str[0] == '\033' && str[1] == '[' && str[str.size() - 1] == 'R')
00233   {
00234     std::size_t found = str.find(';', 2);
00235     if(found != std::string::npos)
00236     {
00237       m_Type               = Type::Cursor;
00238       m_container.m_Cursor = Cursor(static_cast<std::uint16_t>(std::stoi(str.substr(2, found - 2))),
      static_cast<std::uint16_t>(std::stoi(str.substr(found + 1, str.size() - (found + 2)))));
00239     }
00240   }
00241   else if(str[0] == '\033' && str[1] == '[' && str[2] == '<')
00242   {
00243     static std::chrono::time_point<std::chrono::system_clock> old;
00244     bool                                                      not_too_long{false};
00245     if(std::chrono::system_clock::now() - old <= std::chrono::milliseconds{120}) not_too_long = true;
00246     m_Type = Type::Mouse;
00247     std::size_t              pos{3};
00248     std::size_t              pos2{3};
00249     std::vector<std::size_t> values;
00250     while((pos = str.find(';', pos)) != std::string::npos)
00251     {
00252       values.push_back(std::stoull(str.substr(pos2, pos - pos2)));
00253       pos++;
00254       pos2 = pos;
00255     }
00256     values.push_back(std::stoull(str.substr(pos2, str.size() - pos2 - 1)));
00257     static Term::Mouse   first;
00258     static Term::Mouse   second;
00259     Term::Button::Action action;
00260     if(str[str.size() - 1] == 'm') action = Term::Button::Action::Released;
00261     else
00262       action = Term::Button::Action::Pressed;
00263     Term::Button::Type type;
00264     switch(values[0])
00265     {
00266       case 0:
00267       {
00268         type = Term::Button::Type::Right;
00269         break;
00270       }
00271       case 1:
00272       {
00273         type = Term::Button::Type::Wheel;
00274         break;
00275       }
00276       case 2:
00277       {
00278         type = Term::Button::Type::Left;
00279         break;
00280       }
00281       case 35:
00282       {
00283         type   = Term::Button::Type::None;
```

```
00284         action = Term::Button::Action::None;
00285         break;
00286       }
00287       case 64:
00288       {
00289         type   = Term::Button::Type::Wheel;
00290         action = Term::Button::Action::RolledUp;
00291         break;
00292       }
00293       case 65:
00294       {
00295         type   = Term::Button::Type::Wheel;
00296         action = Term::Button::Action::RolledDown;
00297         break;
00298       }
00299       default: break;
00300     }
00301     if(not_too_long && first.row() == second.row() && second.row() == values[1] && first.column() ==
      second.column() && second.column() == values[2] && first.getButton().type() ==
      second.getButton().type() && second.getButton().type() == type && first.getButton().action() ==
      Button::Action::Released && second.getButton().action() == Button::Action::Pressed && action ==
      Button::Action::Pressed) action = Term::Button::Action::DoubleClicked;
00302     second              = first;
00303     first               = Term::Mouse(Term::Button(type, action), values[1], values[2]);
00304     m_container.m_Mouse = first;
00305     old                 = std::chrono::system_clock::now();
00306   }
00307   else if(str.size() <= 10)
00308   {
00309     //https://invisible-island.net/xterm/ctlseqs/ctlseqs.html
00310     // CSI = ESC[ SS3 = ESCO
00311     /*
00312      * Key             Normal     Application
00313      * -------------+----------+-------------
00314      * Cursor Up     | CSI A    | SS3 A
00315      * Cursor Down   | CSI B    | SS3 B
00316      * Cursor Right  | CSI C    | SS3 C
00317      * Cursor Left   | CSI D    | SS3 D
00318      * -------------+----------+-------------
00319      * Key             Normal/Application
00320      * -------------+--------------------
00321      * Cursor Up     | ESC A
00322      * Cursor Down   | ESC B
00323      * Cursor Right  | ESC C
00324      * Cursor Left   | ESC D
00325      * -------------+--------------------
00326     */
00327     if((str == "\u001bOA") || (str == "\u001b[A") || (str == "\u001bA")) { m_container.m_Key =
      Key(Term::Key::ArrowUp); }
00328     else if((str == "\u001bOB") || (str == "\u001b[B") || (str == "\u001bB")) { m_container.m_Key =
      Key(Term::Key::ArrowDown); }
00329     else if((str == "\u001bOC") || (str == "\u001b[C") || (str == "\u001bC")) { m_container.m_Key =
      Key(Term::Key::ArrowRight); }
00330     else if((str == "\u001bOD") || (str == "\u001b[D") || (str == "\u001bD")) { m_container.m_Key =
      Key(Term::Key::ArrowLeft); }
00331     /*
00332      * Key         Normal     Application
00333      * ---------+----------+-------------
00334      * Home      | CSI H    | SS3 H
00335      * End       | CSI F    | SS3 F
00336      * ---------+----------+-------------
00337     */
00338     else if((str == "\u001bOH") || (str == "\u001b[H"))
00339       m_container.m_Key = Key(Term::Key::Home);
00340     else if(str == "\u001bOF" || str == "\u001b[F")
00341       m_container.m_Key = Key(Term::Key::End);
00342     /*
00343      * Key         Escape Sequence
00344      * ---------+-----------------
00345      * F1        | SS3 P
00346      * F2        | SS3 Q
00347      * F3        | SS3 R
00348      * F4        | SS3 S
00349      * F1        | CSI 1 1 ~
00350      * F2        | CSI 1 2 ~
00351      * F3        | CSI 1 3 ~
00352      * F4        | CSI 1 4 ~
00353      * F5        | CSI 1 5 ~
00354      * F6        | CSI 1 7 ~
00355      * F7        | CSI 1 8 ~
00356      * F8        | CSI 1 9 ~
00357      * F9        | CSI 2 0 ~
00358      * F10       | CSI 2 1 ~
00359      * F11       | CSI 2 3 ~
00360      * F12       | CSI 2 4 ~
00361      * ---------+-----------------
00362     */
```

```
00363     else if(str == "\u001bOP" || str == "\u001b[11~")
00364       m_container.m_Key = Key(Term::Key::F1);
00365     else if(str == "\u001bOQ" || str == "\u001b[12~")
00366       m_container.m_Key = Key(Term::Key::F2);
00367     else if(str == "\u001bOR" || str == "\u001b[13~")
00368       m_container.m_Key = Key(Term::Key::F3);
00369     else if(str == "\u001bOS" || str == "\u001b[14~")
00370       m_container.m_Key = Key(Term::Key::F4);
00371     else if(str == "\u001b[15~")
00372       m_container.m_Key = Key(Term::Key::F5);
00373     else if(str == "\u001b[17~")
00374       m_container.m_Key = Key(Term::Key::F6);
00375     else if(str == "\u001b[18~")
00376       m_container.m_Key = Key(Term::Key::F7);
00377     else if(str == "\u001b[19~")
00378       m_container.m_Key = Key(Term::Key::F8);
00379     else if(str == "\u001b[20~")
00380       m_container.m_Key = Key(Term::Key::F9);
00381     else if(str == "\u001b[21~")
00382       m_container.m_Key = Key(Term::Key::F10);
00383     else if(str == "\u001b[23~")
00384       m_container.m_Key = Key(Term::Key::F11);
00385     else if(str == "\u001b[24~")
00386       m_container.m_Key = Key(Term::Key::F12);
00387     /*
00388      * Key         Normal     Application
00389      * ---------+----------+-------------
00390      * Insert    | CSI 2 ~  | CSI 2 ~
00391      * Delete    | CSI 3 ~  | CSI 3 ~
00392      * Home      | CSI 1 ~  | CSI 1 ~
00393      * End       | CSI 4 ~  | CSI 4 ~
00394      * PageUp    | CSI 5 ~  | CSI 5 ~
00395      * PageDown  | CSI 6 ~  | CSI 6 ~
00396      * ---------+----------+-------------
00397     */
00398     else if(str == "\u001b[2~") { m_container.m_Key = Key(Term::Key::Insert); }
00399     else if(str == "\u001b[3~") { m_container.m_Key = Key(Term::Key::Del); }
00400     else if(str == "\u001b[1~") { m_container.m_Key = Key(Term::Key::Home); }
00401     else if(str == "\u001b[4~") { m_container.m_Key = Key(Term::Key::End); }
00402     else if(str == "\u001b[5~") { m_container.m_Key = Key(Term::Key::PageUp); }
00403     else if(str == "\u001b[6~") { m_container.m_Key = Key(Term::Key::PageDown); }
00404     /*
00405      * Key         Escape Sequence
00406      * ---------+-----------------
00407      * F13       | CSI 2 5 ~
00408      * F14       | CSI 2 6 ~
00409      * F15       | CSI 2 8 ~
00410      * F16       | CSI 2 9 ~
00411      * F17       | CSI 3 1 ~
00412      * F18       | CSI 3 2 ~
00413      * F19       | CSI 3 3 ~
00414      * F20       | CSI 3 4 ~
00415      * ---------+-----------------
00416     */
00417     else if(str == "\u001b[25~")
00418       m_container.m_Key = Key(Term::Key::F13);
00419     else if(str == "\u001b[26~")
00420       m_container.m_Key = Key(Term::Key::F14);
00421     else if(str == "\u001b[28~")
00422       m_container.m_Key = Key(Term::Key::F15);
00423     else if(str == "\u001b[29~")
00424       m_container.m_Key = Key(Term::Key::F16);
00425     else if(str == "\u001b[31~")
00426       m_container.m_Key = Key(Term::Key::F17);
00427     else if(str == "\u001b[32~")
00428       m_container.m_Key = Key(Term::Key::F18);
00429     else if(str == "\u001b[33~")
00430       m_container.m_Key = Key(Term::Key::F19);
00431     else if(str == "\u001b[34~")
00432       m_container.m_Key = Key(Term::Key::F20);
00433     else if(str == "\u001b[G")
00434       m_container.m_Key = Key(Term::Key::Value::Numeric5);
00435     else if(Term::Private::is_valid_utf8_code_unit(str))
00436       m_container.m_Key = Key(static_cast<Term::Key::Value>(Term::Private::utf8_to_utf32(str)[0]));
00437     else
00438     {
00439       m_Type = Type::CopyPaste;
00440       new(&this->m_container.m_string) std::string(str);
00441       return;
00442     }
00443     m_Type = Type::Key;
00444   }
00445   else
00446   {
00447     m_Type = Type::CopyPaste;
00448     new(&this->m_container.m_string) std::string(str);
00449   }
```

```
00450 }
```

type()

```
Term::Event::Type Term::Event::type ( ) const
```

Definition at line 203 of file event.cpp.

```
00203 { return m_Type; }
```

8.11.5 Member Data Documentation

m_container

```
container Term::Event::m_container  [private]
```

Definition at line 90 of file event.hpp.

m_Type

```
Type Term::Event::m_Type {Type::Empty}  [private]
```

Definition at line 89 of file event.hpp.

```
00089 {Type::Empty};
```

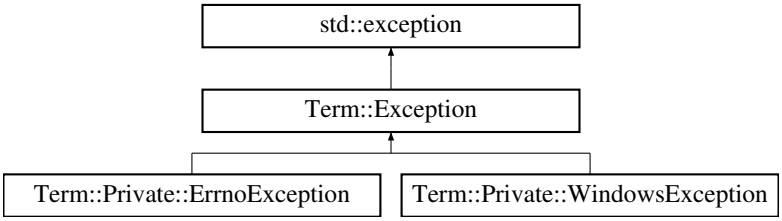
The documentation for this class was generated from the following files:

- cpp-terminal/event.hpp
- cpp-terminal/event.cpp

8.12 Term::Exception Class Reference

```
#include <cpp-terminal/exception.hpp>
```

Inheritance diagram for Term::Exception:



Public Member Functions

- Exception (const std::string &message) noexcept
- Exception (const std::int64_t &code, const std::string &message) noexcept
- Exception (const Exception &)=default
- Exception (Exception &&)=default
- Exception & operator= (Exception &&)=default
- Exception & operator= (const Exception &)=default
- const char * what () const noexcept override
- std::int64_t code () const noexcept
- std::string message () const noexcept
- std::string context () const noexcept
- ~Exception () noexcept override=default

Protected Member Functions

- Exception (const std::int64_t &code) noexcept
- virtual void build_what () const noexcept
- void setMessage (const std::string &message) noexcept
- void setContext (const std::string &context) noexcept
- void setWhat (const std::string &what) const noexcept

Static Protected Attributes

- static const constexpr std::size_t m_maxSize {256}

Private Attributes

- std::int64_t m_code {0}
- std::string m_message
- std::string m_context
- std::string m_what

8.12.1 Detailed Description

Definition at line 19 of file exception.hpp.

8.12.2 Constructor & Destructor Documentation

Exception() [1/5]

```
Term::Exception::Exception (
            const std::string & message )  [explicit], [noexcept]
```

Definition at line 35 of file exception.cpp.

```
00035 : m_message(message) {}
```

Exception() [2/5]

```
Term::Exception::Exception (
            const std::int64_t & code,
            const std::string & message )  [noexcept]
```

Definition at line 37 of file exception.cpp.

```
00037 : m_code(code), m_message(message) {}
```

Exception() [3/5]

```
Term::Exception::Exception (
            const Exception &  )  [default]
```

Exception() [4/5]

```
Term::Exception::Exception (
            Exception &&  )  [default]
```

~Exception()

```
Term::Exception::~Exception ( )  [override], [default], [noexcept]
```

Exception() [5/5]

```
Term::Exception::Exception (
            const std::int64_t & code )  [explicit], [protected], [noexcept]
```

Definition at line 51 of file exception.cpp.

```
00051 : m_code(code) {}
```

8.12.3 Member Function Documentation

build_what()

```
void Term::Exception::build_what ( ) const  [protected], [virtual], [noexcept]
```

Reimplemented in Term::Private::WindowsException, and Term::Private::ErrnoException.

Definition at line 53 of file exception.cpp.

```
00054 {
00055   if(0 == m_code) { m_what = m_message; }
00056   else { m_what = "error " + std::to_string(m_code) + ": " + m_message; }
00057 }
```

code()

```
std::int64_t Term::Exception::code ( ) const  [noexcept]
```

Definition at line 45 of file exception.cpp.

```
00045 { return m_code; }
```

context()

```
std::string Term::Exception::context ( ) const  [noexcept]
```

Definition at line 49 of file exception.cpp.

```
00049 { return m_context; }
```

message()

```
std::string Term::Exception::message ( ) const  [noexcept]
```

Definition at line 47 of file exception.cpp.

```
00047 { return m_message; }
```

operator=() [1/2]

```
Exception & Term::Exception::operator= (
            const Exception &  )  [default]
```

operator=() [2/2]

```
Exception & Term::Exception::operator= (
            Exception &&  )  [default]
```

setContext()

```
void Term::Exception::setContext (
            const std::string & context )  [protected], [noexcept]
```

Definition at line 61 of file exception.cpp.

```
00061 { m_context = context; }
```

setMessage()

```
void Term::Exception::setMessage (
            const std::string & message )  [protected], [noexcept]
```

Definition at line 59 of file exception.cpp.

```
00059 { m_message = message; }
```

setWhat()

```
void Term::Exception::setWhat (
            const std::string & what ) const  [protected], [noexcept]
```

Definition at line 63 of file exception.cpp.

```
00063 { m_what = what; }
```

what()

```
const char * Term::Exception::what ( ) const  [override], [noexcept]
```

Definition at line 39 of file exception.cpp.

```
00040 {
00041   build_what();
00042   return m_what.c_str();
00043 }
```

8.12.4 Member Data Documentation

m_code

```
std::int64_t Term::Exception::m_code {0}  [private]
```

Definition at line 44 of file exception.hpp.

```
00044 {0};
```

m_context

```
std::string Term::Exception::m_context  [private]
```

Definition at line 46 of file exception.hpp.

m_maxSize

```
const constexpr std::size_t Term::Exception::m_maxSize {256}  [static], [constexpr], [protected]
```

Definition at line 41 of file exception.hpp.

```
00041 {256};
```

m_message

```
std::string Term::Exception::m_message  [private]
```

Definition at line 45 of file exception.hpp.

m_what

```
std::string Term::Exception::m_what  [mutable], [private]
```

Definition at line 47 of file exception.hpp.

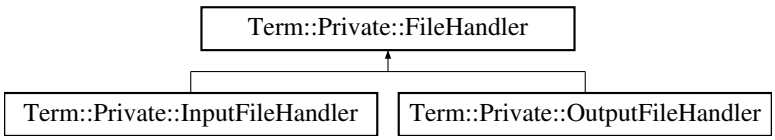
The documentation for this class was generated from the following files:

- cpp-terminal/exception.hpp
- cpp-terminal/private/exception.cpp

8.13 Term::Private::FileHandler Class Reference

`#include <cpp-terminal/private/file.hpp>`

Inheritance diagram for Term::Private::FileHandler:



Public Types

- using Handle = void*

Public Member Functions

- FileHandler (std::recursive_mutex &mutex, const std::string &file, const std::string &mode) noexcept
- FileHandler (const FileHandler &)=delete
- FileHandler (FileHandler &&)=delete
- FileHandler & operator= (const FileHandler &)=delete
- FileHandler & operator= (FileHandler &&)=delete
- virtual ~FileHandler () noexcept
- Handle handle () const noexcept
- bool null () const noexcept
- std::FILE * file () const noexcept
- std::int32_t fd () const noexcept
- void lockIO ()
- void unlockIO ()
- void flush ()

Private Attributes

- std::recursive_mutex & m_mutex
- bool m_null {false}
- Handle m_handle {nullptr}
- FILE * m_file {nullptr}
- std::int32_t m_fd {-1}

8.13.1 Detailed Description

Definition at line 25 of file file.hpp.

8.13.2 Member Typedef Documentation

Handle

```
using Term::Private::FileHandler::Handle = void*
```

Definition at line 29 of file file.hpp.

8.13.3 Constructor & Destructor Documentation

FileHandler() [1/3]

```
Term::Private::FileHandler::FileHandler (
            std::recursive_mutex & mutex,
            const std::string & file,
            const std::string & mode )  [noexcept]
```

Definition at line 43 of file file.cpp.

```
00044     : m_mutex(mutex)
00045 {
00046 #if defined(_WIN32)
00047   m_handle = {CreateFile(file.c_str(), GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ |
      FILE_SHARE_WRITE, nullptr, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, nullptr)};
00048   if(m_handle == INVALID_HANDLE_VALUE)
00049   {
00050     Term::Private::WindowsError().check_if((m_handle = CreateFile("NUL", GENERIC_READ | GENERIC_WRITE,
      FILE_SHARE_READ | FILE_SHARE_WRITE, nullptr, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, nullptr)) ==
      INVALID_HANDLE_VALUE).throw_exception("Problem opening NUL");
00051     m_null = true;
00052   }
00053   Term::Private::Errno().check_if((m_fd = _open_osfhandle(reinterpret_cast<intptr_t>(m_handle),
      _O_RDWR)) == -1).throw_exception("_open_osfhandle(reinterpret_cast<intptr_t>(m_handle), _O_RDWR)");
00054   Term::Private::Errno().check_if(nullptr == (m_file = _fdopen(m_fd,
      mode.c_str()))).throw_exception("_fdopen(m_fd, mode.c_str())");
00055 #else
00056   std::size_t flag{O_ASYNC | O_DSYNC | O_NOCTTY | O_SYNC | O_NDELAY};
00057   flag &= ~static_cast<std::size_t>(O_NONBLOCK);
00058   if(mode.find('r') != std::string::npos) { flag |= O_RDONLY; }
      //NOLINT(abseil-string-find-str-contains)
00059   else if(mode.find('w') != std::string::npos) { flag |= O_WRONLY; }
      //NOLINT(abseil-string-find-str-contains)
00060   else { flag |= O_RDWR; }
00061   m_fd = {::open(file.c_str(), static_cast<int>(flag))};
      //NOLINT(cppcoreguidelines-pro-type-vararg,hicpp-vararg)
00062   if(m_fd == -1)
00063   {
00064     Term::Private::Errno().check_if((m_fd = ::open("/dev/null", static_cast<int>(flag))) ==
      -1).throw_exception(R"(::open("/dev/null", flag))");
      //NOLINT(cppcoreguidelines-pro-type-vararg,hicpp-vararg)
00065     m_null = true;
00066   }
00067   Term::Private::Errno().check_if(nullptr == (m_file = ::fdopen(m_fd,
      mode.c_str()))).throw_exception("::fdopen(m_fd, mode.c_str())");
00068   m_handle = m_file;
00069 #endif
00070   Term::Private::Errno().check_if(std::setvbuf(m_file, nullptr, _IONBF, 0) !=
      0).throw_exception("std::setvbuf(m_file, nullptr, _IONBF, 0)");
00071 }
00072 catch(...)
00073 {
00074   ExceptionHandler(ExceptionDestination::StdErr);
00075 }
```

FileHandler() [2/3]

```
Term::Private::FileHandler::FileHandler (
            const FileHandler &  )  [delete]
```

FileHandler() [3/3]

```
Term::Private::FileHandler::FileHandler (
            FileHandler &&  )  [delete]
```

~FileHandler()

```
Term::Private::FileHandler::~FileHandler ( )  [virtual], [noexcept]
```

Definition at line 77 of file file.cpp.

```
00079 {
00080   flush();
00081   Term::Private::Errno().check_if(0 != std::fclose(m_file)).throw_exception("std::fclose(m_file)");
      //NOLINT(cppcoreguidelines-owning-memory)
00082 }
00083 catch(...)
00084 {
00085   ExceptionHandler(ExceptionDestination::StdErr);
00086 }
```

8.13.4 Member Function Documentation

fd()

```
std::int32_t Term::Private::FileHandler::fd ( ) const  [noexcept]
```

Definition at line 92 of file file.cpp.

```
00092 { return m_fd; }
```

file()

```
std::FILE * Term::Private::FileHandler::file ( ) const  [noexcept]
```

Definition at line 90 of file file.cpp.

```
00090 { return m_file; }
```

flush()

```
void Term::Private::FileHandler::flush ( )
```

Definition at line 142 of file file.cpp.

```
00142 { Term::Private::Errno().check_if(0 != std::fflush(m_file)).throw_exception("std::fflush(m_file)"); }
```

handle()

```
Term::Private::FileHandler::Handle Term::Private::FileHandler::handle ( ) const  [noexcept]
```

Definition at line 94 of file file.cpp.

```
00094 { return m_handle; }
```

lockIO()

```
void Term::Private::FileHandler::lockIO ( )
```

Definition at line 144 of file file.cpp.

```
00144 { m_mutex.lock(); }
```

null()

```
bool Term::Private::FileHandler::null ( ) const  [noexcept]
```

Definition at line 88 of file file.cpp.

```
00088 { return m_null; }
```

operator=() [1/2]

```
FileHandler & Term::Private::FileHandler::operator= (
            const FileHandler &  )  [delete]
```

operator=() [2/2]

```
FileHandler & Term::Private::FileHandler::operator= (
            FileHandler &&  )  [delete]
```

unlockIO()

```
void Term::Private::FileHandler::unlockIO ( )
```

Definition at line 145 of file file.cpp.

```
00145 { m_mutex.unlock(); }
```

8.13.5 Member Data Documentation

m_fd

```
std::int32_t Term::Private::FileHandler::m_fd {-1}  [private]
```

Definition at line 53 of file file.hpp.

```
00053 {-1};
```

m_file

```
FILE* Term::Private::FileHandler::m_file {nullptr}  [private]
```

Definition at line 52 of file file.hpp.

```
00052 {nullptr};
```

m_handle

```
Handle Term::Private::FileHandler::m_handle {nullptr}  [private]
```

Definition at line 51 of file file.hpp.

```
00051 {nullptr};
```

m_mutex

```
std::recursive_mutex& Term::Private::FileHandler::m_mutex  [private]
```

Definition at line 49 of file file.hpp.

m_null

```
bool Term::Private::FileHandler::m_null {false}  [private]
```

Definition at line 50 of file file.hpp.
```
00050 {false};
```

The documentation for this class was generated from the following files:

- cpp-terminal/private/file.hpp
- cpp-terminal/private/file.cpp

8.14 Term::Private::FileInitializer Class Reference

```
#include <cpp-terminal/private/file_initializer.hpp>
```

Public Member Functions

- ~FileInitializer () noexcept
- FileInitializer () noexcept
- FileInitializer (FileInitializer &&)=delete
- FileInitializer (const FileInitializer &)=delete
- FileInitializer & operator= (const FileInitializer &)=delete
- FileInitializer & operator= (FileInitializer &&)=delete

Static Private Member Functions

- static void attachConsole () noexcept

  Attach the console.
- static void detachConsole () noexcept

  Detach the console.
- static void openStandardStreams () noexcept

  Open the standard streams.

Static Private Attributes

- static bool m_consoleCreated = {false}
- static std::size_t m_counter = {0}

8.14.1 Detailed Description

Definition at line 20 of file file_initializer.hpp.

8.14.2 Constructor & Destructor Documentation

~FileInitializer()

```
Term::Private::FileInitializer::~FileInitializer ( )  [noexcept]
```

Definition at line 89 of file file_initializer.cpp.

```
00091 {
00092   --m_counter;
00093   if(0 == m_counter)
00094   {
00095     (&Term::Private::in)->~InputFileHandler();    //NOSONAR(S3432)
00096     (&Term::Private::out)->~OutputFileHandler();  //NOSONAR(S3432)
00097     detachConsole();
00098   }
00099 }
00100 catch(...)
00101 {
00102   ExceptionHandler(ExceptionDestination::StdErr);
00103 }
```

FileInitializer() [1/3]

```
Term::Private::FileInitializer::FileInitializer ( )  [noexcept]
```

Definition at line 70 of file file_initializer.cpp.

```
00072 {
00073   // MacOS was not happy wish a static mutex in the class so we create it and pass to each class;
00074   static std::recursive_mutex ioMutex;
00075   if(0 == m_counter)
00076   {
00077     attachConsole();
00078     openStandardStreams();
00079     if(nullptr == new(&Term::Private::in) InputFileHandler(ioMutex)) { throw
      Term::Exception("new(&Term::Private::in) InputFileHandler(ioMutex)"); }
00080     if(nullptr == new(&Term::Private::out) OutputFileHandler(ioMutex)) { throw
      Term::Exception("new(&Term::Private::out) OutputFileHandler(ioMutex)"); }
00081   }
00082   ++m_counter;
00083 }
00084 catch(...)
00085 {
00086   ExceptionHandler(ExceptionDestination::StdErr);
00087 }
```

FileInitializer() [2/3]

```
Term::Private::FileInitializer::FileInitializer (
            FileInitializer &&  )  [delete]
```

FileInitializer() [3/3]

```
Term::Private::FileInitializer::FileInitializer (
            const FileInitializer &  )  [delete]
```

8.14.3 Member Function Documentation

attachConsole()

```
void Term::Private::FileInitializer::attachConsole ( )  [static], [private], [noexcept]
```

Attach the console.

Check if a console is attached to the process. If not, try to attach to the console. If there is no console, then create one.

Definition at line 31 of file file_initializer.cpp.

```
00033 {
00034 #if defined(_WIN32)
00035   // If something happen here we still don't have a console so we can only use a MessageBox to warn
      the users something is very bad and that they should contact us.
00036   Term::Private::WindowsError
      error{Term::Private::WindowsError().check_if(AttachConsole(ATTACH_PARENT_PROCESS) == 0)};
00037   bool                        need_allocation{false};
00038   switch(error.error())
00039   {
00040     case ERROR_SUCCESS: break;
      // Has been attached
00041     case ERROR_ACCESS_DENIED: need_allocation = false; break;
      // Already attached that's good !
00042     case ERROR_INVALID_PARAMETER: error.throw_exception("The specified process does not exist !");
      break;  // Should never happen !
00043     case ERROR_INVALID_HANDLE: need_allocation = true; break;
      // Need to allocate th console !
00044   }
00045   if(need_allocation)
00046   {
00047     Term::Private::WindowsError().check_if(AllocConsole() == 0).throw_exception("AllocConsole()");
00048     m_consoleCreated = true;
00049   }
00050 #endif
00051 }
00052 catch(...)
00053 {
00054   detachConsole();
00055   ExceptionHandler(ExceptionDestination::MessageBox);
00056 }
```

detachConsole()

```
void Term::Private::FileInitializer::detachConsole ( )  [static], [private], [noexcept]
```

Detach the console.

If a console as been created, then delete it.

Definition at line 58 of file file_initializer.cpp.

```
00060 {
00061 #if defined(_WIN32)
00062   if(m_consoleCreated) { Term::Private::WindowsError().check_if(0 ==
      FreeConsole()).throw_exception("FreeConsole()"); }
00063 #endif
00064 }
00065 catch(...)
00066 {
00067   ExceptionHandler(ExceptionDestination::MessageBox);
00068 }
```

openStandardStreams()

```
void Term::Private::FileInitializer::openStandardStreams ( )  [static], [private], [noexcept]
```

Open the standard streams.

Open **stdout stderr stdin** and adjust their buffer size and line discipline.

Definition at line 105 of file file_initializer.cpp.

```
00107 {
00108 #if defined(_WIN32)
00109   FILE* fDummy{nullptr};
00110   if(_fileno(stderr) < 0 || _get_osfhandle(_fileno(stderr)) < 0) {
      Term::Private::Errno().check_if(_wfreopen_s(&fDummy, L"CONOUT$", L"w", stderr) !=
      0).throw_exception(R"(_wfreopen_s(&fDummy, L"CONOUT$", L"w", stderr))"); }
00111   if(_fileno(stdout) < 0 || _get_osfhandle(_fileno(stdout)) < 0) {
      Term::Private::Errno().check_if(_wfreopen_s(&fDummy, L"CONOUT$", L"w", stdout) !=
      0).throw_exception(R"(_wfreopen_s(&fDummy, L"CONOUT$", L"w", stdout))"); }
00112   if(_fileno(stdin) < 0 || _get_osfhandle(_fileno(stdin)) < 0) {
      Term::Private::Errno().check_if(_wfreopen_s(&fDummy, L"CONIN$", L"r", stdin) !=
      0).throw_exception(R"(_wfreopen_s(&fDummy, L"CONIN$", L"r", stdin))"); }
00113   const std::size_t bestSize{BUFSIZ > 4096 ? BUFSIZ : 4096};
00114 #else
00115   if(::fileno(stderr) < 0) { Term::Private::Errno().check_if(nullptr == std::freopen("/dev/tty", "w",
      stderr)).throw_exception(R"(std::freopen("/dev/tty", "w", stderr))"); }
      //NOLINT(cppcoreguidelines-owning-memory)
00116   if(::fileno(stdout) < 0) { Term::Private::Errno().check_if(nullptr == std::freopen("/dev/tty", "w",
      stdout)).throw_exception(R"(std::freopen("/dev/tty", "w", stdout))"); }
      //NOLINT(cppcoreguidelines-owning-memory)
00117   if(::fileno(stdin) < 0) { Term::Private::Errno().check_if(nullptr == std::freopen("/dev/tty", "r",
      stdin)).throw_exception(R"(std::freopen("/dev/tty", "r", stdin))"); }
      //NOLINT(cppcoreguidelines-owning-memory)
00118   struct stat stats = {};
00119   ::stat("/dev/tty", &stats);
00120   const std::size_t bestSize{static_cast<std::size_t>(stats.st_blksize) > 0 ?
      static_cast<std::size_t>(stats.st_blksize) : BUFSIZ};  //NOSONAR(S1774)
00121 #endif
00122   Term::Private::Errno().check_if(std::setvbuf(stderr, nullptr, _IONBF, 0) !=
      0).throw_exception("std::setvbuf(stderr, nullptr, _IONBF, 0)");
00123   Term::Private::Errno().check_if(std::setvbuf(stdout, nullptr, _IOLBF, bestSize) !=
      0).throw_exception("std::setvbuf(stdout, nullptr, _IOLBF, bestSize)");
00124   Term::Private::Errno().check_if(std::setvbuf(stdin, nullptr, _IOLBF, bestSize) !=
      0).throw_exception("std::setvbuf(stdin, nullptr, _IOLBF, bestSize)");
00125 }
00126 catch(...)
00127 {
00128   ExceptionHandler(ExceptionDestination::StdErr);
00129 }
```

operator=() [1/2]

```
FileInitializer & Term::Private::FileInitializer::operator= (
            const FileInitializer &  )  [delete]
```

operator=() [2/2]

```
FileInitializer & Term::Private::FileInitializer::operator= (
            FileInitializer &&  )  [delete]
```

8.14.4 Member Data Documentation

m_consoleCreated

```
bool Term::Private::FileInitializer::m_consoleCreated = {false}  [static], [private]
```

Definition at line 27 of file file_initializer.hpp.

```
00030 :
00031   static bool        m_consoleCreated;
00032   static std::size_t m_counter;
00033
00039   static void attachConsole() noexcept;
00040
00046   static void detachConsole() noexcept;
00047
00053   static void openStandardStreams() noexcept;
00054 };
00055
00056 }  // namespace Private
00057
00058 }  // namespace Term
```

m_counter

```
std::size_t Term::Private::FileInitializer::m_counter = {0}  [static], [private]
```

Definition at line 29 of file file_initializer.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/private/file_initializer.hpp
- cpp-terminal/private/file_initializer.cpp

8.15 Term::Focus Class Reference

Class to return the focus of the terminal.

```
#include <cpp-terminal/focus.hpp>
```

Public Types

- enum class Type : std::int8_t { Unknown = -1 , Out = 0 , In = 1 }

Public Member Functions

- Focus ()=default
- Focus (const Term::Focus::Type &type)
- Term::Focus::Type type () const

  Get the type of focus.
- bool in () const

  *Check is the focus is **in**.*
- bool out () const

  *Check is the focus is **out**.*
- bool operator== (const Term::Focus &focus) const
- bool operator!= (const Term::Focus &focus) const

Private Attributes

- Term::Focus::Type m_focus {Term::Focus::Type::Unknown}

8.15.1 Detailed Description

Class to return the focus of the terminal.

Definition at line 21 of file focus.hpp.

8.15.2 Member Enumeration Documentation

Type

```
enum class Term::Focus::Type :  std::int8_t  [strong]
```

Enumerator

Unknown	The terminal focus is **unknown**.
Out	The terminal focus is **out**.
In	The terminal focus is **in**.

Definition at line 24 of file focus.hpp.

```
00025   {
00026     Unknown = -1,
00027     Out     = 0,
00028     In      = 1,
00029   };
```

8.15.3 Constructor & Destructor Documentation

Focus() [1/2]

```
Term::Focus::Focus ( )  [default]
```

Focus() [2/2]

```
Term::Focus::Focus (
            const Term::Focus::Type & type )  [explicit]
```

Definition at line 15 of file focus.cpp.

```
00015 : m_focus(type) {}
```

8.15.4 Member Function Documentation

in()

```
bool Term::Focus::in ( ) const
```

Check is the focus is **in**.

Returns

true : The terminal has focus **in**.

false : The terminal has focus **out**.

Definition at line 19 of file focus.cpp.

```
00019 { return m_focus == Term::Focus::Type::In; }
```

operator"!=()

```
bool Term::Focus::operator!= (
            const Term::Focus & focus ) const
```

Definition at line 25 of file focus.cpp.

```
00025 { return !(*this == focus); }
```

operator==()

```
bool Term::Focus::operator== (
            const Term::Focus & focus ) const
```

Definition at line 23 of file focus.cpp.

```
00023 { return m_focus == focus.m_focus; }
```

out()

```
bool Term::Focus::out ( ) const
```

Check is the focus is **out**.

Returns

true : The terminal has focus **out**.

false : The terminal has focus **in**.

Definition at line 21 of file focus.cpp.

```
00021 { return m_focus == Term::Focus::Type::Out; }
```

type()

```
Term::Focus::Type Term::Focus::type ( ) const
```

Get the type of focus.

Returns

Term::Focus::Type

Definition at line 17 of file focus.cpp.

```
00017 { return m_focus; }
```

8.15.5 Member Data Documentation

m_focus

```
Term::Focus::Type Term::Focus::m_focus {Term::Focus::Type::Unknown}  [private]
```

Definition at line 62 of file focus.hpp.

```
00062 {Term::Focus::Type::Unknown};
```

The documentation for this class was generated from the following files:

- cpp-terminal/focus.hpp
- cpp-terminal/focus.cpp

8.16 Term::Private::Input Class Reference

```
#include <cpp-terminal/private/input.hpp>
```

Public Member Functions

- Input ()

Static Public Member Functions

- static void startReading ()
- static Term::Event getEvent ()
- static Term::Event getEventBlocking ()

Static Private Member Functions

- static void read_event ()
- static void read_raw ()
- static void read_windows_key (const std::uint16_t &virtual_key_code, const std::uint32_t &control_key_state, const std::size_t &occurrence)
- static void init_thread ()

Static Private Attributes

- static std::thread m_thread = std::thread(Term::Private::Input::read_event)
- static Term::Private::BlockingQueue m_events
- static int m_poll {-1}

8.16.1 Detailed Description

Definition at line 25 of file input.hpp.

8.16.2 Constructor & Destructor Documentation

Input()

```
Term::Private::Input::Input ( )
```

Definition at line 305 of file input.cpp.

```
00305 {}
```

8.16.3 Member Function Documentation

getEvent()

```
Term::Event Term::Private::Input::getEvent ( )  [static]
```

Definition at line 317 of file input.cpp.

```
00317 { return m_events.pop(); }
```

getEventBlocking()

```
Term::Event Term::Private::Input::getEventBlocking ( )  [static]
```

Definition at line 319 of file input.cpp.

```
00320 {
00321   static std::mutex                   cv_m;
00322   static std::unique_lock<std::mutex> lk(cv_m);
00323   if(m_events.empty()) m_events.wait_for_events(lk);
00324   return m_events.pop();
00325 }
```

init_thread()

```
void Term::Private::Input::init_thread ( )  [static], [private]
```

Definition at line 84 of file input.cpp.

```
00085 {
00086   Term::Private::Sigwinch::unblockSigwinch();
00087 #if defined(__linux__)
00088   m_poll = {::epoll_create1(EPOLL_CLOEXEC)};
00089   ::epoll_event signal;
00090   signal.events  = {EPOLLIN};
00091   signal.data.fd = {Term::Private::Sigwinch::get()};
00092   ::epoll_ctl(m_poll, EPOLL_CTL_ADD, Term::Private::Sigwinch::get(), &signal);
00093   ::epoll_event input;
00094   input.events  = {EPOLLIN};
00095   input.data.fd = {Term::Private::in.fd()};
00096   ::epoll_ctl(m_poll, EPOLL_CTL_ADD, Term::Private::in.fd(), &input);
00097 #endif
00098 }
```

read_event()

```
void Term::Private::Input::read_event ( )  [static], [private]
```

Definition at line 100 of file input.cpp.

```
00101 {
00102   init_thread();
00103   while(true)
00104   {
00105 #if defined(_WIN32)
00106     WaitForSingleObject(Term::Private::in.handle(), INFINITE);
00107     read_raw();
00108 #elif defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00109     if(Term::Private::Sigwinch::isSigwinch()) m_events.push(screen_size());
00110     read_raw();
00111 #else
00112     ::epoll_event ret;
00113     if(epoll_wait(m_poll, &ret, 1, -1) == 1)
00114     {
00115       if(Term::Private::Sigwinch::isSigwinch(ret.data.fd)) m_events.push(Term::Screen(screen_size()));
00116       else
00117         read_raw();
00118     }
00119 #endif
00120   }
00121 }
```

read_raw()

```
void Term::Private::Input::read_raw ( )  [static], [private]
```

Definition at line 203 of file input.cpp.

```
00204 {
00205 #ifdef _WIN32
00206   DWORD to_read{0};
00207   GetNumberOfConsoleInputEvents(Private::in.handle(), &to_read);
00208   if(to_read == 0) return;
00209   DWORD                     read{0};
00210   std::vector<INPUT_RECORD> events{to_read};
00211   if(!ReadConsoleInputW(Private::in.handle(), &events[0], to_read, &read) || read != to_read)
      Term::Exception("ReadFile() failed");
00212   std::wstring ret;
00213   bool         need_windows_size{false};
00214   for(std::size_t i = 0; i != read; ++i)
00215   {
00216     switch(events[i].EventType)
00217     {
00218       case KEY_EVENT:
00219       {
00220         if(events[i].Event.KeyEvent.bKeyDown)
00221         {
00222           if(events[i].Event.KeyEvent.uChar.UnicodeChar == 0)
      read_windows_key(events[i].Event.KeyEvent.wVirtualKeyCode, events[i].Event.KeyEvent.dwControlKeyState,
      events[i].Event.KeyEvent.wRepeatCount);
00223           else
00224             ret.append(events[i].Event.KeyEvent.wRepeatCount,
      events[i].Event.KeyEvent.uChar.UnicodeChar == Term::Key::Del ?
      static_cast<wchar_t>(Key(Term::Key::Value::Backspace)) :
      static_cast<wchar_t>(events[i].Event.KeyEvent.uChar.UnicodeChar));
00225         }
00226         break;
00227       }
00228       case FOCUS_EVENT:
00229       {
00230         sendString(m_events, ret);
00231
      m_events.push(Event(Focus(static_cast<Term::Focus::Type>(events[i].Event.FocusEvent.bSetFocus))));
00232         break;
00233       }
00234       case MENU_EVENT:
00235       {
00236         sendString(m_events, ret);
00237         break;
00238       }
00239       case MOUSE_EVENT:
00240       {
00241         sendString(m_events, ret);
00242         static MOUSE_EVENT_RECORD old_state;
```

```
00243         if(events[i].Event.MouseEvent.dwEventFlags == MOUSE_WHEELED ||
      events[i].Event.MouseEvent.dwEventFlags == MOUSE_HWHEELED)
00244           ;
00245         else if(old_state.dwButtonState == events[i].Event.MouseEvent.dwButtonState &&
      old_state.dwMousePosition.X == events[i].Event.MouseEvent.dwMousePosition.X &&
      old_state.dwMousePosition.Y == events[i].Event.MouseEvent.dwMousePosition.Y && old_state.dwEventFlags
      == events[i].Event.MouseEvent.dwEventFlags)
00246           break;
00247         std::int32_t state{static_cast<std::int32_t>(events[i].Event.MouseEvent.dwButtonState)};
00248         switch(events[i].Event.MouseEvent.dwEventFlags)
00249         {
00250           case 0:
00251           {
00252             m_events.push(Term::Mouse(setButton(static_cast<std::int32_t>(old_state.dwButtonState),
      state), static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00253             ;
00254             break;
00255           }
00256           case MOUSE_MOVED:
00257           {
00258             m_events.push(Term::Mouse(setButton(static_cast<std::int32_t>(old_state.dwButtonState),
      state), static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00259             ;
00260             break;
00261           }
00262           case DOUBLE_CLICK:
00263           {
00264
      m_events.push(Term::Mouse(Term::Button(setButton(static_cast<std::int32_t>(old_state.dwButtonState),
      state).type(), Term::Button::Action::DoubleClicked),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00265             break;
00266           }
00267           case MOUSE_WHEELED:
00268           {
00269             if(state > 0) m_events.push(Term::Mouse(Button(Term::Button::Type::Wheel,
      Term::Button::Action::RolledUp),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00270             else
00271               m_events.push(Term::Mouse(Button(Term::Button::Type::Wheel,
      Term::Button::Action::RolledDown),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00272             break;
00273           }
00274           case MOUSE_HWHEELED:
00275           {
00276             if(state > 0) m_events.push(Term::Mouse(Button(Term::Button::Type::Wheel,
      Term::Button::Action::ToRight),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00277             else
00278               m_events.push(Term::Mouse(Button(Term::Button::Type::Wheel,
      Term::Button::Action::ToLeft),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00279             break;
00280           }
00281           default: break;
00282         }
00283         old_state = events[i].Event.MouseEvent;
00284         break;
00285       }
00286       case WINDOW_BUFFER_SIZE_EVENT:
00287       {
00288         need_windows_size = true;  // if we send directly it's too much generations
00289         sendString(m_events, ret);
00290         break;
00291       }
00292       default: break;
00293     }
00294   }
00295   sendString(m_events, ret);
00296   if(need_windows_size == true) { m_events.push(screen_size()); }
00297 #else
00298   Private::in.lockIO();
00299   std::string ret = Term::Private::in.read();
00300   Private::in.unlockIO();
00301   if(!ret.empty()) m_events.push(Event(ret.c_str()));
00302 #endif
00303 }
```

read_windows_key()

```
void Term::Private::Input::read_windows_key (
            const std::uint16_t & virtual_key_code,
            const std::uint32_t & control_key_state,
            const std::size_t & occurrence )  [static], [private]
```

Definition at line 124 of file input.cpp.

```
00125 {
00126   // First check if we have ALT etc (CTRL is already done so skip it)
00127   Term::MetaKey toAdd{Term::MetaKey::Value::None};
00128   if(((control_key_state & LEFT_ALT_PRESSED) == LEFT_ALT_PRESSED) || ((control_key_state &
      RIGHT_ALT_PRESSED) == RIGHT_ALT_PRESSED)) toAdd += Term::MetaKey::Value::Alt;
00129   if(((control_key_state & LEFT_CTRL_PRESSED) == LEFT_CTRL_PRESSED) || ((control_key_state &
      RIGHT_CTRL_PRESSED) == RIGHT_CTRL_PRESSED)) toAdd += Term::MetaKey::Value::Ctrl;
00130
00131   switch(virtual_key_code)
00132   {
00133     case VK_CANCEL:  //??
00134     case VK_CLEAR:   //??
00135     case VK_SHIFT:
00136     case VK_CONTROL:
00137     case VK_MENU:
00138     case VK_PAUSE:  //??
00139     case VK_CAPITAL:
00140     case VK_KANA:  //??
00141     //case VK_HANGUL: // Same
00142     case VK_JUNJA:  // ?
00143     case VK_FINAL:  // ?
00144     case VK_HANJA:
00145     //case VK_KANJI: // Same
00146     case VK_CONVERT:     // ?
00147     case VK_NONCONVERT:  // ?
00148     case VK_ACCEPT:      // ?
00149     case VK_MODECHANGE:  // ?
00150       break;
00151     case VK_PRIOR: m_events.push(std::move(toAdd + Term::Key(Key::Value::PageUp)), occurrence); break;
00152     case VK_NEXT: m_events.push(std::move(toAdd + Term::Key(Key::Value::PageDown)), occurrence);
      break;
00153     case VK_END: m_events.push(std::move(toAdd + Term::Key(Key::Value::End)), occurrence); break;
00154     case VK_HOME: m_events.push(std::move(toAdd + Term::Key(Key::Value::Home)), occurrence); break;
00155     case VK_LEFT: m_events.push(std::move(toAdd + Term::Key(Key::Value::ArrowLeft)), occurrence);
      break;
00156     case VK_UP: m_events.push(std::move(toAdd + Term::Key(Key::Value::ArrowUp)), occurrence); break;
00157     case VK_RIGHT: m_events.push(std::move(toAdd + Term::Key(Key::Value::ArrowRight)), occurrence);
      break;
00158     case VK_DOWN: m_events.push(std::move(toAdd + Term::Key(Key::Value::ArrowDown)), occurrence);
      break;
00159     case VK_SELECT:   //?
00160     case VK_PRINT:    //?
00161     case VK_EXECUTE:  //?
00162       break;
00163     case VK_SNAPSHOT: m_events.push(std::move(toAdd + Term::Key(Key::Value::PrintScreen)),
      occurrence); break;
00164     case VK_INSERT: m_events.push(std::move(toAdd + Term::Key(Key::Value::Insert)), occurrence);
      break;
00165     case VK_DELETE: m_events.push(std::move(toAdd + Term::Key(Key::Value::Del)), occurrence); break;
00166     case VK_HELP:   //?
00167     case VK_LWIN:   //Maybe allow to detect Windows key Left and right
00168     case VK_RWIN:   //Maybe allow to detect Windows key Left and right
00169     case VK_APPS:   //?
00170     case VK_SLEEP:  //?
00171       break;
00172     case VK_F1: m_events.push(std::move(toAdd + Term::Key(Key::Value::F1)), occurrence); break;
00173     case VK_F2: m_events.push(std::move(toAdd + Term::Key(Key::Value::F2)), occurrence); break;
00174     case VK_F3: m_events.push(std::move(toAdd + Term::Key(Key::Value::F3)), occurrence); break;
00175     case VK_F4: m_events.push(std::move(toAdd + Term::Key(Key::Value::F4)), occurrence); break;
00176     case VK_F5: m_events.push(std::move(toAdd + Term::Key(Key::Value::F5)), occurrence); break;
00177     case VK_F6: m_events.push(std::move(toAdd + Term::Key(Key::Value::F6)), occurrence); break;
00178     case VK_F7: m_events.push(std::move(toAdd + Term::Key(Key::Value::F7)), occurrence); break;
00179     case VK_F8: m_events.push(std::move(toAdd + Term::Key(Key::Value::F8)), occurrence); break;
00180     case VK_F9: m_events.push(std::move(toAdd + Term::Key(Key::Value::F9)), occurrence); break;
00181     case VK_F10: m_events.push(std::move(toAdd + Term::Key(Key::Value::F10)), occurrence); break;
00182     case VK_F11: m_events.push(std::move(toAdd + Term::Key(Key::Value::F11)), occurrence); break;
00183     case VK_F12: m_events.push(std::move(toAdd + Term::Key(Key::Value::F12)), occurrence); break;
00184     case VK_F13: m_events.push(std::move(toAdd + Term::Key(Key::Value::F13)), occurrence); break;
00185     case VK_F14: m_events.push(std::move(toAdd + Term::Key(Key::Value::F14)), occurrence); break;
00186     case VK_F15: m_events.push(std::move(toAdd + Term::Key(Key::Value::F15)), occurrence); break;
00187     case VK_F16: m_events.push(std::move(toAdd + Term::Key(Key::Value::F16)), occurrence); break;
00188     case VK_F17: m_events.push(std::move(toAdd + Term::Key(Key::Value::F17)), occurrence); break;
00189     case VK_F18: m_events.push(std::move(toAdd + Term::Key(Key::Value::F18)), occurrence); break;
00190     case VK_F19: m_events.push(std::move(toAdd + Term::Key(Key::Value::F19)), occurrence); break;
00191     case VK_F20: m_events.push(std::move(toAdd + Term::Key(Key::Value::F20)), occurrence); break;
```

```
00192     case VK_F21: m_events.push(std::move(toAdd + Term::Key(Key::Value::F21)), occurrence); break;
00193     case VK_F22: m_events.push(std::move(toAdd + Term::Key(Key::Value::F22)), occurrence); break;
00194     case VK_F23: m_events.push(std::move(toAdd + Term::Key(Key::Value::F23)), occurrence); break;
00195     case VK_F24: m_events.push(std::move(toAdd + Term::Key(Key::Value::F24)), occurrence); break;
00196     case VK_NUMLOCK:
00197     case VK_SCROLL:
00198     default: break;
00199   }
00200 }
```

startReading()

```
void Term::Private::Input::startReading ( )  [static]
```

Definition at line 307 of file input.cpp.

```
00308 {
00309   static bool activated{false};
00310   if(!activated)
00311   {
00312     m_thread.detach();
00313     activated = true;
00314   }
00315 }
```

8.16.4 Member Data Documentation

m_events

```
Term::Private::BlockingQueue Term::Private::Input::m_events  [static], [private]
```

Definition at line 41 of file input.hpp.

m_poll

```
int Term::Private::Input::m_poll {-1}  [static], [private]
```

Definition at line 82 of file input.hpp.

m_thread

```
std::thread Term::Private::Input::m_thread = std::thread(Term::Private::Input::read_event)
[static], [private]
```

Definition at line 40 of file input.hpp.

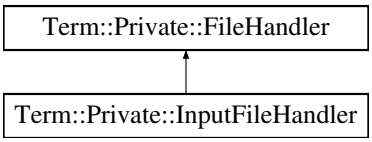
The documentation for this class was generated from the following files:

- cpp-terminal/private/input.hpp
- cpp-terminal/private/input.cpp

8.17 Term::Private::InputFileHandler Class Reference

`#include <cpp-terminal/private/file.hpp>`

Inheritance diagram for Term::Private::InputFileHandler:



Public Member Functions

- InputFileHandler (std::recursive_mutex &io_mutex) noexcept
- InputFileHandler (const InputFileHandler &)=delete
- InputFileHandler (InputFileHandler &&)=delete
- InputFileHandler & operator= (InputFileHandler &&)=delete
- InputFileHandler & operator= (const InputFileHandler &)=delete
- ~InputFileHandler () override=default
- std::string read () const

Public Member Functions inherited from Term::Private::FileHandler

- FileHandler (std::recursive_mutex &mutex, const std::string &file, const std::string &mode) noexcept
- FileHandler (const FileHandler &)=delete
- FileHandler (FileHandler &&)=delete
- FileHandler & operator= (const FileHandler &)=delete
- FileHandler & operator= (FileHandler &&)=delete
- virtual ~FileHandler () noexcept
- Handle handle () const noexcept
- bool null () const noexcept
- std::FILE * file () const noexcept
- std::int32_t fd () const noexcept
- void lockIO ()
- void unlockIO ()
- void flush ()

Static Public Attributes

- static const constexpr char * m_file {"CONIN$"}

Additional Inherited Members

Public Types inherited from Term::Private::FileHandler

- using Handle = void*

8.17.1 Detailed Description

Definition at line 75 of file file.hpp.

8.17.2 Constructor & Destructor Documentation

InputFileHandler() [1/3]

```
Term::Private::InputFileHandler::InputFileHandler (
            std::recursive_mutex & io_mutex )  [explicit], [noexcept]
```

Definition at line 157 of file file.cpp.

```
00158     : FileHandler(io_mutex, m_file, "r")
00159 {
00160   //noop
00161 }
00162 catch(...)
00163 {
00164   ExceptionHandler(ExceptionDestination::StdErr);
00165 }
```

InputFileHandler() [2/3]

```
Term::Private::InputFileHandler::InputFileHandler (
            const InputFileHandler &  )  [delete]
```

InputFileHandler() [3/3]

```
Term::Private::InputFileHandler::InputFileHandler (
            InputFileHandler &&  )  [delete]
```

~InputFileHandler()

```
Term::Private::InputFileHandler::~InputFileHandler ( )  [override], [default]
```

8.17.3 Member Function Documentation

operator=() [1/2]

```
InputFileHandler & Term::Private::InputFileHandler::operator= (
            const InputFileHandler &  )  [delete]
```

operator=() [2/2]

```
InputFileHandler & Term::Private::InputFileHandler::operator= (
            InputFileHandler &&  )  [delete]
```

read()

```
std::string Term::Private::InputFileHandler::read ( ) const
```

Definition at line 122 of file file.cpp.

```
00123 {
00124 #if defined(_WIN32)
00125   DWORD       nread{0};
00126   std::string ret(4096, '\0');
00127   errno = 0;
00128   ReadConsole(Private::in.handle(), &ret[0], static_cast<DWORD>(ret.size()), &nread, nullptr);
00129   return ret.c_str();
00130 #else
00131   std::size_t nread{0};
00132   Term::Private::Errno().check_if(::ioctl(Private::in.fd(), FIONREAD, &nread) !=
      0).throw_exception("::ioctl(Private::in.fd(), FIONREAD, &nread)");
      //NOLINT(cppcoreguidelines-pro-type-vararg,hicpp-vararg)
00133   std::string ret(nread, '\0');
00134   if(nread != 0)
00135   {
00136     Term::Private::Errno().check_if(::read(Private::in.fd(), &ret[0], ret.size()) ==
      -1).throw_exception("::read(Private::in.fd(), &ret[0], ret.size())");
      //NOLINT(readability-container-data-pointer)
00137   }
00138   return ret;
00139 #endif
00140 }
```

8.17.4 Member Data Documentation

m_file

```
const constexpr char* Term::Private::InputFileHandler::m_file {"CONIN$"}  [static], [constexpr]
```

Definition at line 87 of file file.hpp.

```
00087 {"CONIN$"};
```

The documentation for this class was generated from the following files:

- cpp-terminal/private/file.hpp
- cpp-terminal/private/file.cpp

8.18 Term::IOStreamInitializer Class Reference

```
#include <cpp-terminal/iostream_initializer.hpp>
```

Public Member Functions

- ~IOStreamInitializer () noexcept
- IOStreamInitializer () noexcept
- IOStreamInitializer (const IOStreamInitializer &)=delete
- IOStreamInitializer (IOStreamInitializer &&)=delete
- IOStreamInitializer & operator= (IOStreamInitializer &&)=delete
- IOStreamInitializer & operator= (const IOStreamInitializer &)=delete

Static Private Attributes

- static std::size_t m_counter {0}

8.18.1 Detailed Description

Definition at line 17 of file iostream_initializer.hpp.

8.18.2 Constructor & Destructor Documentation

~IOStreamInitializer()

```
Term::IOStreamInitializer::~IOStreamInitializer ( )  [noexcept]
```

Definition at line 43 of file iostream_initializer.cpp.

```
00045 {
00046   --m_counter;
00047   if(0 == m_counter)
00048   {
00049     (&Term::cout)->~TOstream();
00050     (&Term::cerr)->~TOstream();
00051     (&Term::clog)->~TOstream();
00052     (&Term::cin)->~TIstream();
00053   }
00054 }
00055 catch(...)
00056 {
00057   ExceptionHandler(Private::ExceptionDestination::StdErr);
00058 }
```

IOStreamInitializer() [1/3]

```
Term::IOStreamInitializer::IOStreamInitializer ( )  [noexcept]
```

Definition at line 23 of file iostream_initializer.cpp.

```
00025 {
00026   if(0 == m_counter)
00027   {
00028     static const std::ios_base::Init       iostreams_init;  // Init std::cout etc...
00029     static const Term::TerminalInitializer terminal_init;   // Make sure terminal is set up.
00030     new(&Term::cout) TOstream(Term::Buffer::Type::FullBuffered, BUFSIZ);
00031     new(&Term::clog) TOstream(Term::Buffer::Type::LineBuffered, BUFSIZ);
00032     new(&Term::cerr) TOstream(Term::Buffer::Type::Unbuffered, 0);
00033     new(&Term::cin) TIstream(Term::Buffer::Type::FullBuffered, BUFSIZ);
00034     if(is_stdin_a_tty()) { std::cin.rdbuf(Term::cin.rdbuf()); }
00035   }
00036   ++m_counter;
00037 }
00038 catch(...)
00039 {
00040   ExceptionHandler(Private::ExceptionDestination::StdErr);
00041 }
```

IOStreamInitializer() [2/3]

```
Term::IOStreamInitializer::IOStreamInitializer (
            const IOStreamInitializer &  )  [delete]
```

IOStreamInitializer() [3/3]

```
Term::IOStreamInitializer::IOStreamInitializer (
            IOStreamInitializer &&  )  [delete]
```

8.18.3 Member Function Documentation

operator=() [1/2]

```
IOStreamInitializer & Term::IOStreamInitializer::operator= (
            const IOStreamInitializer &  )  [delete]
```

operator=() [2/2]

```
IOStreamInitializer & Term::IOStreamInitializer::operator= (
            IOStreamInitializer &&  )  [delete]
```

8.18.4 Member Data Documentation

m_counter

```
std::size_t Term::IOStreamInitializer::m_counter {0}  [static], [private]
```

Definition at line 21 of file iostream_initializer.hpp.

```
00027 :
00028   static std::size_t m_counter;
00029 };
00030
00031 }  // namespace Term
```

The documentation for this class was generated from the following files:

- cpp-terminal/iostream_initializer.hpp
- cpp-terminal/iostream_initializer.cpp

8.19 Term::Key Class Reference

```
#include <cpp-terminal/key.hpp>
```

Public Types

- enum Value : std::int32_t {
  NoKey = -1 , Ctrl_Arobase = 0 , Ctrl_A = 1 , Ctrl_B = 2 ,
  Ctrl_C = 3 , Ctrl_D = 4 , Ctrl_E = 5 , Ctrl_F = 6 ,
  Ctrl_G = 7 , Ctrl_H = 8 , Ctrl_I = 9 , Ctrl_J = 10 ,
  Ctrl_K = 11 , Ctrl_L = 12 , Ctrl_M = 13 , Ctrl_N = 14 ,
  Ctrl_O = 15 , Ctrl_P = 16 , Ctrl_Q = 17 , Ctrl_R = 18 ,
  Ctrl_S = 19 , Ctrl_T = 20 , Ctrl_U = 21 , Ctrl_V = 22 ,
  Ctrl_W = 23 , Ctrl_X = 24 , Ctrl_Y = 25 , Ctrl_Z = 26 ,
  Ctrl_OpenBracket = 27 , Ctrl_BackSlash = 28 , Ctrl_CloseBracket = 29 , Ctrl_Caret = 30 ,
  Ctrl_Underscore = 31 , Space = 32 , ExclamationMark = 33 , Quote = 34 ,
  Hash = 35 , Dollar = 36 , Percent = 37 , Ampersand = 38 ,
  Apostrophe = 39 , OpenParenthesis = 40 , CloseParenthesis = 41 , Asterisk = 42 ,
  Plus = 43 , Comma = 44 , Hyphen = 45 , Minus = 45 ,
  Period = 46 , Slash = 47 , Zero = 48 , One = 49 ,
  Two = 50 , Three = 51 , Four = 52 , Five = 53 ,

Six = 54 , Seven = 55 , Eight = 56 , Nine = 57 ,
Colon = 58 , Semicolon = 59 , LessThan = 60 , OpenChevron = 60 ,
Equal = 61 , GreaterThan = 62 , CloseChevron = 62 , QuestionMark = 63 ,
Arobase = 64 , A = 65 , B = 66 , C = 67 ,
D = 68 , E = 69 , F = 70 , G = 71 ,
H = 72 , I = 73 , J = 74 , K = 75 ,
L = 76 , M = 77 , N = 78 , O = 79 ,
P = 80 , Q = 81 , R = 82 , S = 83 ,
T = 84 , U = 85 , V = 86 , W = 87 ,
X = 88 , Y = 89 , Z = 90 , OpenBracket = 91 ,
Backslash = 92 , CloseBracket = 93 , Caret = 94 , Underscore = 95 ,
GraveAccent = 96 , a = 97 , b = 98 , c = 99 ,
d = 100 , e = 101 , f = 102 , g = 103 ,
h = 104 , i = 105 , j = 106 , k = 107 ,
l = 108 , m = 109 , n = 110 , o = 111 ,
p = 112 , q = 113 , r = 114 , s = 115 ,
t = 116 , u = 117 , v = 118 , w = 119 ,
x = 120 , y = 121 , z = 122 , OpenBrace = 123 ,
VerticalBar = 124 , Close_Brace = 125 , Tilde = 126 , CTRL_QuestionMark = 127 ,
Null = 0 , Backspace = 8 , Tab = 9 , Enter = 13 ,
Esc = 27 , Del = 127 , ArrowLeft = 0x10FFFF + 1 , ArrowRight = 0x10FFFF + 2 ,
ArrowUp = 0x10FFFF + 3 , ArrowDown = 0x10FFFF + 4 , Numeric5 = 0x10FFFF + 5 , Home = 0x10FFFF + 6 ,
Insert = 0x10FFFF + 7 , End = 0x10FFFF + 8 , PageUp = 0x10FFFF + 9 , PageDown = 0x10FFFF + 10 ,
F1 = 0x10FFFF + 11 , F2 = 0x10FFFF + 12 , F3 = 0x10FFFF + 13 , F4 = 0x10FFFF + 14 ,
F5 = 0x10FFFF + 15 , F6 = 0x10FFFF + 16 , F7 = 0x10FFFF + 17 , F8 = 0x10FFFF + 18 ,
F9 = 0x10FFFF + 19 , F10 = 0x10FFFF + 20 , F11 = 0x10FFFF + 21 , F12 = 0x10FFFF + 22 ,
F13 = 0x10FFFF + 23 , F14 = 0x10FFFF + 24 , F15 = 0x10FFFF + 25 , F16 = 0x10FFFF + 26 ,
F17 = 0x10FFFF + 27 , F18 = 0x10FFFF + 28 , F19 = 0x10FFFF + 29 , F20 = 0x10FFFF + 30 ,
F21 = 0x10FFFF + 31 , F22 = 0x10FFFF + 32 , F23 = 0x10FFFF + 33 , F24 = 0x10FFFF + 34 ,
PrintScreen = 0x10FFFF + 35 , Menu = 0x10FFFF + 36 }

- using value_type = std::int32_t

Public Member Functions

- constexpr Key ()
- constexpr Key (const Key &key)=default
- Key & operator= (const Key &key)=default
- constexpr Key (const Value &v)
- Key & operator= (const Value &v)
- constexpr Key (char val)
- Key & operator= (char val)
- constexpr Key (std::int32_t val)
- Key & operator= (std::int32_t val)
- constexpr Key (std::size_t val)
- Key & operator= (std::size_t val)
- constexpr Key (char32_t val)
- Key & operator= (char32_t val)
- constexpr operator std::int32_t () const
- constexpr bool iscntrl () const
- constexpr bool isblank () const
- constexpr bool isspace () const
- constexpr bool isupper () const
- constexpr bool islower () const
- constexpr bool isalpha () const
- constexpr bool isdigit () const

- constexpr bool isxdigit () const
- constexpr bool isalnum () const
- constexpr bool ispunct () const
- constexpr bool isgraph () const
- constexpr bool isprint () const
- constexpr bool isunicode () const
- constexpr Key tolower () const
- constexpr Key toupper () const
- constexpr bool isASCII () const
- constexpr bool isExtendedASCII () const
- constexpr bool hasCtrlAlt () const
- constexpr bool hasCtrl () const
- constexpr bool hasAlt () const
- constexpr bool empty () const
- void append_name (std::string &strOut) const
- std::string name () const
- std::string str () const

Public Attributes

- std::int32_t value

Friends

- constexpr bool operator== (Key l, Key r)
- constexpr bool operator== (Key l, char r)
- constexpr bool operator== (char l, Key r)
- constexpr bool operator== (Key l, char32_t r)
- constexpr bool operator== (char32_t l, Key r)
- constexpr bool operator== (Key l, std::int32_t r)
- constexpr bool operator== (std::int32_t l, Key r)
- constexpr bool operator== (Key l, std::size_t r)
- constexpr bool operator== (std::size_t l, Key r)
- constexpr bool operator!= (Key l, Key r)
- constexpr bool operator!= (Key l, char r)
- constexpr bool operator!= (char l, Key r)
- constexpr bool operator!= (Key l, char32_t r)
- constexpr bool operator!= (char32_t l, Key r)
- constexpr bool operator!= (Key l, std::int32_t r)
- constexpr bool operator!= (std::int32_t l, Key r)
- constexpr bool operator!= (Key l, std::size_t r)
- constexpr bool operator!= (std::size_t l, Key r)
- constexpr bool operator< (Key l, Key r)
- constexpr bool operator< (Key l, char r)
- constexpr bool operator< (char l, Key r)
- constexpr bool operator< (Key l, char32_t r)
- constexpr bool operator< (char32_t l, Key r)
- constexpr bool operator< (Key l, std::int32_t r)
- constexpr bool operator< (std::int32_t l, Key r)
- constexpr bool operator< (Key l, std::size_t r)
- constexpr bool operator< (std::size_t l, Key r)
- constexpr bool operator>= (Key l, Key r)
- constexpr bool operator>= (Key l, char r)

- constexpr bool operator>= (char l, Key r)
- constexpr bool operator>= (Key l, char32_t r)
- constexpr bool operator>= (char32_t l, Key r)
- constexpr bool operator>= (Key l, std::int32_t r)
- constexpr bool operator>= (std::int32_t l, Key r)
- constexpr bool operator>= (Key l, std::size_t r)
- constexpr bool operator>= (std::size_t l, Key r)
- constexpr bool operator> (Key l, Key r)
- constexpr bool operator> (Key l, char r)
- constexpr bool operator> (char l, Key r)
- constexpr bool operator> (Key l, char32_t r)
- constexpr bool operator> (char32_t l, Key r)
- constexpr bool operator> (Key l, std::int32_t r)
- constexpr bool operator> (std::int32_t l, Key r)
- constexpr bool operator> (Key l, std::size_t r)
- constexpr bool operator> (std::size_t l, Key r)
- constexpr bool operator<= (Key l, Key r)
- constexpr bool operator<= (Key l, char r)
- constexpr bool operator<= (char l, Key r)
- constexpr bool operator<= (Key l, char32_t r)
- constexpr bool operator<= (char32_t l, Key r)
- constexpr bool operator<= (Key l, std::int32_t r)
- constexpr bool operator<= (std::int32_t l, Key r)
- constexpr bool operator<= (Key l, std::size_t r)
- constexpr bool operator<= (std::size_t l, Key r)

8.19.1 Detailed Description

Definition at line 85 of file key.hpp.

8.19.2 Member Typedef Documentation

value_type

```
using Term::Key::value_type = std::int32_t
```

Definition at line 88 of file key.hpp.

8.19.3 Member Enumeration Documentation

Value

```
enum Term::Key::Value :  std::int32_t
```

Enumerator

NoKey	
Ctrl_Arobase	
Ctrl_A	
Ctrl_B	

Enumerator

Ctrl_C	
Ctrl_D	
Ctrl_E	
Ctrl_F	
Ctrl_G	
Ctrl_H	
Ctrl_I	
Ctrl_J	
Ctrl_K	
Ctrl_L	
Ctrl_M	
Ctrl_N	
Ctrl_O	
Ctrl_P	
Ctrl_Q	
Ctrl_R	
Ctrl_S	
Ctrl_T	
Ctrl_U	
Ctrl_V	
Ctrl_W	
Ctrl_X	
Ctrl_Y	
Ctrl_Z	
Ctrl_OpenBracket	
Ctrl_BackSlash	
Ctrl_CloseBracket	
Ctrl_Caret	
Ctrl_Underscore	
Space	
ExclamationMark	
Quote	
Hash	
Dollar	
Percent	
Ampersand	
Apostrophe	
OpenParenthesis	
CloseParenthesis	
Asterisk	
Plus	
Comma	
Hyphen	
Minus	
Period	
Slash	
Zero	
One	
Two	

Enumerator

Three	
Four	
Five	
Six	
Seven	
Eight	
Nine	
Colon	
Semicolon	
LessThan	
OpenChevron	
Equal	
GreaterThan	
CloseChevron	
QuestionMark	
Arobase	
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	
M	
N	
O	
P	
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	
OpenBracket	
Backslash	
CloseBracket	
Caret	
Underscore	
GraveAccent	
a	
b	
c	

Enumerator

d	
e	
f	
g	
h	
i	
j	
k	
l	
m	
n	
o	
p	
q	
r	
s	
t	
u	
v	
w	
x	
y	
z	
OpenBrace	
VerticalBar	
Close_Brace	
Tilde	
CTRL_QuestionMark	
Null	
Backspace	
Tab	
Enter	
Esc	
Del	
ArrowLeft	
ArrowRight	
ArrowUp	
ArrowDown	
Numeric5	
Home	
Insert	
End	
PageUp	
PageDown	
F1	
F2	
F3	
F4	
F5	
F6	
F7	

Enumerator

F8	
F9	
F10	
F11	
F12	
F13	
F14	
F15	
F16	
F17	
F18	
F19	
F20	
F21	
F22	
F23	
F24	
PrintScreen	
Menu	

Definition at line 90 of file key.hpp.

```
00091   {
00092     NoKey = -1,
00093     // Now use « to for detecting special key + key press
00094
00095     // Begin ASCII (some ASCII names has been change to their Ctrl_+key part) the value is different
      to be able to do
00096     Ctrl_Arobase      = 0,
00097     Ctrl_A            = 1,
00098     Ctrl_B            = 2,
00099     Ctrl_C            = 3,
00100     Ctrl_D            = 4,
00101     Ctrl_E            = 5,
00102     Ctrl_F            = 6,
00103     Ctrl_G            = 7,
00104     Ctrl_H            = 8, /*corresponds to Backspace*/
00105     Ctrl_I            = 9, /*corresponds to Tab*/
00106     Ctrl_J            = 10,
00107     Ctrl_K            = 11,
00108     Ctrl_L            = 12,
00109     Ctrl_M            = 13, /*corresponds to Enter*/
00110     Ctrl_N            = 14,
00111     Ctrl_O            = 15,
00112     Ctrl_P            = 16,
00113     Ctrl_Q            = 17,
00114     Ctrl_R            = 18,
00115     Ctrl_S            = 19,
00116     Ctrl_T            = 20,
00117     Ctrl_U            = 21,
00118     Ctrl_V            = 22,
00119     Ctrl_W            = 23,
00120     Ctrl_X            = 24,
00121     Ctrl_Y            = 25,
00122     Ctrl_Z            = 26,
00123     Ctrl_OpenBracket  = 27, /*corresponds to Escape*/
00124     Ctrl_BackSlash    = 28,
00125     Ctrl_CloseBracket = 29,
00126     Ctrl_Caret        = 30,
00127     Ctrl_Underscore   = 31,
00128     Space             = 32,
00129     ExclamationMark   = 33,
00130     Quote             = 34,
00131     Hash              = 35,
00132     Dollar            = 36,
00133     Percent           = 37,
00134     Ampersand         = 38,
00135     Apostrophe        = 39,
00136     OpenParenthesis   = 40,
00137     CloseParenthesis  = 41,
00138     Asterisk          = 42,
```

```
00139     Plus             = 43,
00140     Comma            = 44,
00141     Hyphen           = 45,
00142     Minus            = 45,
00143     Period           = 46,
00144     Slash            = 47,
00145     Zero             = 48,
00146     One              = 49,
00147     Two              = 50,
00148     Three            = 51,
00149     Four             = 52,
00150     Five             = 53,
00151     Six              = 54,
00152     Seven            = 55,
00153     Eight            = 56,
00154     Nine             = 57,
00155     Colon            = 58,
00156     Semicolon        = 59,
00157     LessThan         = 60,
00158     OpenChevron      = 60,
00159     Equal            = 61,
00160     GreaterThan      = 62,
00161     CloseChevron     = 62,
00162     QuestionMark     = 63,
00163     Arobase          = 64,
00164     A                = 65,
00165     B                = 66,
00166     C                = 67,
00167     D                = 68,
00168     E                = 69,
00169     F                = 70,
00170     G                = 71,
00171     H                = 72,
00172     I                = 73,
00173     J                = 74,
00174     K                = 75,
00175     L                = 76,
00176     M                = 77,
00177     N                = 78,
00178     O                = 79,
00179     P                = 80,
00180     Q                = 81,
00181     R                = 82,
00182     S                = 83,
00183     T                = 84,
00184     U                = 85,
00185     V                = 86,
00186     W                = 87,
00187     X                = 88,
00188     Y                = 89,
00189     Z                = 90,
00190     OpenBracket      = 91,
00191     Backslash        = 92,
00192     CloseBracket     = 93,
00193     Caret            = 94,
00194     Underscore       = 95,
00195     GraveAccent      = 96,
00196     a                = 97,
00197     b                = 98,
00198     c                = 99,
00199     d                = 100,
00200     e                = 101,
00201     f                = 102,
00202     g                = 103,
00203     h                = 104,
00204     i                = 105,
00205     j                = 106,
00206     k                = 107,
00207     l                = 108,
00208     m                = 109,
00209     n                = 110,
00210     o                = 111,
00211     p                = 112,
00212     q                = 113,
00213     r                = 114,
00214     s                = 115,
00215     t                = 116,
00216     u                = 117,
00217     v                = 118,
00218     w                = 119,
00219     x                = 120,
00220     y                = 121,
00221     z                = 122,
00222     OpenBrace        = 123,
00223     VerticalBar      = 124,
00224     Close_Brace      = 125,
00225     Tilde            = 126,
```

```
00226     CTRL_QuestionMark = 127, /*corresponds to DEL*/
00227     // Very useful CTRL_* alternative names
00228     Null              = 0,
00229     Backspace         = 8,
00230     Tab               = 9,
00231     Enter             = 13,
00232     Esc               = 27,
00233     Del               = 127,
00234     //
00235     // End ASCII
00236     // Extended ASCII goes up to 255
00237     // Last Unicode codepage 0x10FFFF
00238     ArrowLeft         = 0x10FFFF + 1,
00239     ArrowRight        = 0x10FFFF + 2,
00240     ArrowUp           = 0x10FFFF + 3,
00241     ArrowDown         = 0x10FFFF + 4,
00242     Numeric5          = 0x10FFFF + 5,
00243     Home              = 0x10FFFF + 6,
00244     Insert            = 0x10FFFF + 7,
00245     End               = 0x10FFFF + 8,
00246     PageUp            = 0x10FFFF + 9,
00247     PageDown          = 0x10FFFF + 10,
00248     F1                = 0x10FFFF + 11,
00249     F2                = 0x10FFFF + 12,
00250     F3                = 0x10FFFF + 13,
00251     F4                = 0x10FFFF + 14,
00252     F5                = 0x10FFFF + 15,
00253     F6                = 0x10FFFF + 16,
00254     F7                = 0x10FFFF + 17,
00255     F8                = 0x10FFFF + 18,
00256     F9                = 0x10FFFF + 19,
00257     F10               = 0x10FFFF + 20,
00258     F11               = 0x10FFFF + 21,
00259     F12               = 0x10FFFF + 22,
00260     F13               = 0x10FFFF + 23,
00261     F14               = 0x10FFFF + 24,
00262     F15               = 0x10FFFF + 25,
00263     F16               = 0x10FFFF + 26,
00264     F17               = 0x10FFFF + 27,
00265     F18               = 0x10FFFF + 28,
00266     F19               = 0x10FFFF + 29,
00267     F20               = 0x10FFFF + 30,
00268     F21               = 0x10FFFF + 31,
00269     F22               = 0x10FFFF + 32,
00270     F23               = 0x10FFFF + 33,
00271     F24               = 0x10FFFF + 34,
00272     PrintScreen       = 0x10FFFF + 35,
00273     Menu              = 0x10FFFF + 36,
00274   };
```

8.19.4 Constructor & Destructor Documentation

Key() [1/7]

```
constexpr Term::Key::Key ( )  [inline], [constexpr]
```

Definition at line 276 of file key.hpp.

```
00276 : value(NoKey) {}
```

Key() [2/7]

```
constexpr Term::Key::Key (
            const Key & key )  [constexpr], [default]
```

Key() [3/7]

```
constexpr Term::Key::Key (
            const Value & v )  [inline], [constexpr]
```

Definition at line 280 of file key.hpp.

```
00280 : value(static_cast<std::int32_t>(v)) {}
```

Key() [4/7]

```
constexpr Term::Key::Key (
            char val )  [inline], [explicit], [constexpr]
```

Definition at line 287 of file key.hpp.

```
00287 : value(static_cast<std::int32_t>(val)) {}
```

Key() [5/7]

```
constexpr Term::Key::Key (
            std::int32_t val )  [inline], [constexpr]
```

Definition at line 294 of file key.hpp.

```
00294 : value(val) {}
```

Key() [6/7]

```
constexpr Term::Key::Key (
            std::size_t val )  [inline], [explicit], [constexpr]
```

Definition at line 301 of file key.hpp.

```
00301 : value(static_cast<std::int32_t>(val)) {}
```

Key() [7/7]

```
constexpr Term::Key::Key (
            char32_t val )  [inline], [explicit], [constexpr]
```

Definition at line 308 of file key.hpp.

```
00308 : value(static_cast<std::int32_t>(val)) {}
```

8.19.5 Member Function Documentation

append_name()

```
void Term::Key::append_name (
            std::string & strOut ) const
```

Definition at line 16 of file key.cpp.

```
00017 {
00018   Term::Key key = *this;
00019   if(key == Term::Key::NoKey) return;
00020   if(key.hasAlt())
00021   {
00022     strOut += "Alt+";
00023     key = static_cast<Term::Key>(key.value - static_cast<std::int32_t>(Term::MetaKey::Value::Alt));
00024   }
00025   if(key.hasCtrl())
00026   {
00027     strOut += "Ctrl+";
00028     if(!key.iscntrl()) key = static_cast<Term::Key>(key.value -
      static_cast<std::int32_t>(Term::MetaKey::Value::Ctrl));
00029   }
00030   if(key == Term::Key::Tab) strOut += "Tab";
00031   else if(key == Term::Key::Enter)
00032     strOut += "Enter";
00033   else if(key == Term::Key::Esc)
```

```
00034     strOut += "Esc";
00035   else if(key == Term::Key::Backspace)
00036     strOut += "Backspace";
00037   else if(key == Term::Key::Del)
00038     strOut += "Del";
00039   else if(key.iscntrl())
00040     strOut += static_cast<char>(key.value + 64);
00041   else if(key == Term::Key::Space)
00042     strOut += "Space";
00043   else if(key.isunicode()) { strOut +=
      Term::Private::utf32_to_utf8(static_cast<char32_t>(this->value)); }
00044   else
00045   {
00046     switch(key)
00047     {
00048       case Term::Key::ArrowLeft: strOut += "Left Arrow"; break;
00049       case Term::Key::ArrowRight: strOut += "Right Arrow"; break;
00050       case Term::Key::ArrowUp: strOut += "Up arrow"; break;
00051       case Term::Key::ArrowDown: strOut += "Down arrow"; break;
00052       case Term::Key::Numeric5: strOut += "5 Numeric pad"; break;
00053       case Term::Key::Home: strOut += "Home"; break;
00054       case Term::Key::Insert: strOut += "Insert"; break;
00055       case Term::Key::End: strOut += "End"; break;
00056       case Term::Key::PageUp: strOut += "Page up"; break;
00057       case Term::Key::PageDown: strOut += "Page down"; break;
00058       case Term::Key::F1: strOut += "F1"; break;
00059       case Term::Key::F2: strOut += "F2"; break;
00060       case Term::Key::F3: strOut += "F3"; break;
00061       case Term::Key::F4: strOut += "F4"; break;
00062       case Term::Key::F5: strOut += "F5"; break;
00063       case Term::Key::F6: strOut += "F6"; break;
00064       case Term::Key::F7: strOut += "F7"; break;
00065       case Term::Key::F8: strOut += "F8"; break;
00066       case Term::Key::F9: strOut += "F9"; break;
00067       case Term::Key::F10: strOut += "F10"; break;
00068       case Term::Key::F11: strOut += "F11"; break;
00069       case Term::Key::F12: strOut += "F12"; break;
00070       case Term::Key::F13: strOut += "F13"; break;
00071       case Term::Key::F14: strOut += "F14"; break;
00072       case Term::Key::F15: strOut += "F15"; break;
00073       case Term::Key::F16: strOut += "F16"; break;
00074       case Term::Key::F17: strOut += "F17"; break;
00075       case Term::Key::F18: strOut += "F18"; break;
00076       case Term::Key::F19: strOut += "F19"; break;
00077       case Term::Key::F20: strOut += "F20"; break;
00078       case Term::Key::F21: strOut += "F21"; break;
00079       case Term::Key::F22: strOut += "F22"; break;
00080       case Term::Key::F23: strOut += "F23"; break;
00081       case Term::Key::F24: strOut += "F24"; break;
00082       case Term::Key::PrintScreen: strOut += "Print Screen"; break;
00083       case Term::Key::Menu: strOut += "Menu"; break;
00084       default: break;
00085     }
00086   }
00087 }
```

empty()

```
constexpr bool Term::Key::empty ( ) const  [inline], [constexpr]
```

Definition at line 412 of file key.hpp.

```
00412 { return (this->value == Key::NoKey); }
```

hasAlt()

```
constexpr bool Term::Key::hasAlt ( ) const  [inline], [constexpr]
```

Definition at line 410 of file key.hpp.

```
00410 { return (this->value & static_cast<std::int32_t>(MetaKey::Value::Alt)) ==
      static_cast<std::int32_t>(MetaKey::Value::Alt); }
```

hasCtrl()

```
constexpr bool Term::Key::hasCtrl ( ) const  [inline], [constexpr]
```

Definition at line 403 of file key.hpp.

```
00404   {
00405     // Need to suppress the TAB etc...
00406     return ((this->iscntrl() || this->hasCtrlAll()) && *this != Key::Backspace && *this != Key::Tab &&
      *this != Key::Esc && *this != Key::Enter && *this != Key::Del);
00407   }
```

hasCtrlAll()

```
constexpr bool Term::Key::hasCtrlAll ( ) const  [inline], [constexpr]
```

Definition at line 400 of file key.hpp.

```
00400 { return this->iscntrl() || ((this->value & static_cast<std::int32_t>(MetaKey::Value::Ctrl)) ==
      static_cast<std::int32_t>(MetaKey::Value::Ctrl)); }
```

isalnum()

```
constexpr bool Term::Key::isalnum ( ) const  [inline], [constexpr]
```

Definition at line 385 of file key.hpp.

```
00385 { return (this->isdigit() || this->isalpha()); }
```

isalpha()

```
constexpr bool Term::Key::isalpha ( ) const  [inline], [constexpr]
```

Definition at line 382 of file key.hpp.

```
00382 { return (this->isupper() || this->islower()); }
```

isASCII()

```
constexpr bool Term::Key::isASCII ( ) const  [inline], [constexpr]
```

Definition at line 394 of file key.hpp.

```
00394 { return *this >= Key::Null && *this <= Key::Del; }
```

isblank()

```
constexpr bool Term::Key::isblank ( ) const  [inline], [constexpr]
```

Definition at line 378 of file key.hpp.

```
00378 { return *this == Key::Tab || *this == Key::Space; }
```

iscntrl()

```
constexpr bool Term::Key::iscntrl ( ) const  [inline], [constexpr]
```

Definition at line 377 of file key.hpp.

```
00377 { return (*this >= Key::Null && *this <= Key::Ctrl_Underscore) || *this == Key::Del; }
```

isdigit()

```
constexpr bool Term::Key::isdigit ( ) const  [inline], [constexpr]
```

Definition at line 383 of file key.hpp.

```
00383 { return *this >= Key::Zero && *this <= Key::Nine; }
```

isExtendedASCII()

```
constexpr bool Term::Key::isExtendedASCII ( ) const  [inline], [constexpr]
```

Definition at line 397 of file key.hpp.

```
00397 { return *this >= Key::Null && this->value <= 255L; }
```

isgraph()

```
constexpr bool Term::Key::isgraph ( ) const  [inline], [constexpr]
```

Definition at line 387 of file key.hpp.

```
00387 { return (this->isalnum() || this->ispunct()); }
```

islower()

```
constexpr bool Term::Key::islower ( ) const  [inline], [constexpr]
```

Definition at line 381 of file key.hpp.

```
00381 { return *this >= Key::a && *this <= Key::z; }
```

isprint()

```
constexpr bool Term::Key::isprint ( ) const  [inline], [constexpr]
```

Definition at line 388 of file key.hpp.

```
00388 { return (this->isgraph() || *this == Key::Space); }
```

ispunct()

```
constexpr bool Term::Key::ispunct ( ) const  [inline], [constexpr]
```

Definition at line 386 of file key.hpp.

```
00386 { return (*this >= Key::ExclamationMark && *this <= Key::Slash) || (*this >= Key::Colon && *this <=
      Key::Arobase) || (*this >= Key::OpenBracket && *this <= Key::GraveAccent) || (*this >= Key::OpenBrace
      && *this <= Key::Tilde); }
```

isspace()

```
constexpr bool Term::Key::isspace ( ) const  [inline], [constexpr]
```

Definition at line 379 of file key.hpp.

```
00379 { return this->isblank() || (*this >= Key::Ctrl_J && *this <= Key::Enter); }
```

isunicode()

```
constexpr bool Term::Key::isunicode ( ) const  [inline], [constexpr]
```

Definition at line 389 of file key.hpp.

```
00389 { return *this >= Key::Null && this->value <= 0x10FFFFL; }
```

isupper()

```
constexpr bool Term::Key::isupper ( ) const  [inline], [constexpr]
```

Definition at line 380 of file key.hpp.

```
00380 { return *this >= Key::A && *this <= Key::Z; }
```

isxdigit()

```
constexpr bool Term::Key::isxdigit ( ) const  [inline], [constexpr]
```

Definition at line 384 of file key.hpp.

```
00384 { return this->isdigit() || (*this >= Key::A && *this <= Key::F) || (*this >= Key::a && *this <=
      Key::f); }
```

name()

```
std::string Term::Key::name ( ) const
```

Definition at line 89 of file key.cpp.

```
00090 {
00091   std::string str;
00092   this->append_name(str);
00093   return str;
00094 }
```

operator std::int32_t()

```
constexpr Term::Key::operator std::int32_t ( ) const  [inline], [constexpr]
```

Definition at line 315 of file key.hpp.

```
00315 { return this->value; }
```

operator=() [1/6]

```
Key & Term::Key::operator= (
            char val )  [inline]
```

Definition at line 288 of file key.hpp.

```
00289   {
00290     value = static_cast<std::int32_t>(val);
00291     return *this;
00292   }
```

operator=() [2/6]

```
Key & Term::Key::operator= (
            char32_t val )  [inline]
```

Definition at line 309 of file key.hpp.

```
00310   {
00311     value = static_cast<std::int32_t>(val);
00312     return *this;
00313   }
```

operator=() [3/6]

```
Key & Term::Key::operator= (
            const Key & key )  [inline], [default]
```

operator=() [4/6]

```
Key & Term::Key::operator= (
            const Value & v )  [inline]
```

Definition at line 281 of file key.hpp.

```
00282   {
00283     this->value = static_cast<std::int32_t>(v);
00284     return *this;
00285   }
```

operator=() [5/6]

```
Key & Term::Key::operator= (
            std::int32_t val )  [inline]
```

Definition at line 295 of file key.hpp.

```
00296   {
00297     value = val;
00298     return *this;
00299   }
```

operator=() [6/6]

```
Key & Term::Key::operator= (
            std::size_t val )  [inline]
```

Definition at line 302 of file key.hpp.

```
00303   {
00304     value = static_cast<std::int32_t>(val);
00305     return *this;
00306   }
```

str()

```
std::string Term::Key::str ( ) const
```

Definition at line 96 of file key.cpp.

```
00096 { return Term::Private::utf32_to_utf8(static_cast<char32_t>(this->value)); }
```

tolower()

```
constexpr Key Term::Key::tolower ( ) const  [inline], [constexpr]
```

Definition at line 390 of file key.hpp.

```
00390 { return (this->isalpha() && this->isupper()) ? Key(this->value + 32) : *this; }
```

toupper()

```
constexpr Key Term::Key::toupper ( ) const  [inline], [constexpr]
```

Definition at line 391 of file key.hpp.

```
00391 { return (this->isalpha() && this->islower()) ? Key(this->value - 32) : *this; }
```

8.19.6 Friends And Related Symbol Documentation

operator"!= [1/9]

```
constexpr bool operator!= (
            char l,
            Key r )  [friend]
```

Definition at line 329 of file key.hpp.

```
00329 { return !(l == r); }
```

operator"!= [2/9]

```
constexpr bool operator!= (
            char32_t l,
            Key r )  [friend]
```

Definition at line 331 of file key.hpp.

```
00331 { return !(l == r); }
```

operator"!= [3/9]

```
constexpr bool operator!= (
            Key l,
            char r )  [friend]
```

Definition at line 328 of file key.hpp.

```
00328 { return !(l == r); }
```

operator"!= [4/9]

```
constexpr bool operator!= (
            Key l,
            char32_t r )  [friend]
```

Definition at line 330 of file key.hpp.

```
00330 { return !(l == r); }
```

operator"!= [5/9]

```
constexpr bool operator!= (
            Key l,
            Key r )  [friend]
```

Definition at line 327 of file key.hpp.

```
00327 { return !(l == r); }
```

operator"!= [6/9]

```
constexpr bool operator!= (
            Key l,
            std::int32_t r )  [friend]
```

Definition at line 332 of file key.hpp.

```
00332 { return !(l == r); }
```

operator"!= [7/9]

```
constexpr bool operator!= (
            Key l,
            std::size_t r )  [friend]
```

Definition at line 334 of file key.hpp.

```
00334 { return !(l == r); }
```

operator"!= [8/9]

```
constexpr bool operator!= (
            std::int32_t l,
            Key r )  [friend]
```

Definition at line 333 of file key.hpp.

```
00333 { return !(l == r); }
```

operator"!= [9/9]

```
constexpr bool operator!= (
            std::size_t l,
            Key r )  [friend]
```

Definition at line 335 of file key.hpp.

```
00335 { return !(l == r); }
```

operator< [1/9]

```
constexpr bool operator< (
            char l,
            Key r )  [friend]
```

Definition at line 339 of file key.hpp.

```
00339 { return Key(l) < r; }
```

operator< [2/9]

```
constexpr bool operator< (
            char32_t l,
            Key r )  [friend]
```

Definition at line 341 of file key.hpp.

```
00341 { return Key(l) < r; }
```

operator< [3/9]

```
constexpr bool operator< (
            Key l,
            char r )  [friend]
```

Definition at line 338 of file key.hpp.

```
00338 { return l < Key(r); }
```

operator< [4/9]

```
constexpr bool operator< (
            Key l,
            char32_t r )  [friend]
```

Definition at line 340 of file key.hpp.

```
00340 { return l < Key(r); }
```

operator< [5/9]

```
constexpr bool operator< (
            Key l,
            Key r )  [friend]
```

Definition at line 337 of file key.hpp.

```
00337 { return l.value < r.value; }
```

operator< [6/9]

```
constexpr bool operator< (
            Key l,
            std::int32_t r )  [friend]
```

Definition at line 342 of file key.hpp.

```
00342 { return l < Key(r); }
```

operator< [7/9]

```
constexpr bool operator< (
            Key l,
            std::size_t r )  [friend]
```

Definition at line 344 of file key.hpp.

```
00344 { return static_cast<std::size_t>(l.value) < r; }
```

operator< [8/9]

```
constexpr bool operator< (
            std::int32_t l,
            Key r )  [friend]
```

Definition at line 343 of file key.hpp.

```
00343 { return Key(l) < r; }
```

operator< [9/9]

```
constexpr bool operator< (
            std::size_t l,
            Key r )  [friend]
```

Definition at line 345 of file key.hpp.

```
00345 { return l < static_cast<std::size_t>(r.value); }
```

operator<= [1/9]

```
constexpr bool operator<= (
            char l,
            Key r )  [friend]
```

Definition at line 369 of file key.hpp.

```
00369 { return !(l > r); }
```

operator<= [2/9]

```
constexpr bool operator<= (
            char32_t l,
            Key r )  [friend]
```

Definition at line 371 of file key.hpp.

```
00371 { return !(l > r); }
```

operator<= [3/9]

```
constexpr bool operator<= (
            Key l,
            char r )  [friend]
```

Definition at line 368 of file key.hpp.

```
00368 { return !(l > r); }
```

operator<= [4/9]

```
constexpr bool operator<= (
            Key l,
            char32_t r )  [friend]
```

Definition at line 370 of file key.hpp.

```
00370 { return !(l > r); }
```

operator<= [5/9]

```
constexpr bool operator<= (
            Key l,
            Key r )  [friend]
```

Definition at line 367 of file key.hpp.

```
00367 { return !(l > r); }
```

operator<= [6/9]

```
constexpr bool operator<= (
            Key l,
            std::int32_t r )  [friend]
```

Definition at line 372 of file key.hpp.

```
00372 { return !(l > r); }
```

operator<= [7/9]

```
constexpr bool operator<= (
            Key l,
            std::size_t r )  [friend]
```

Definition at line 374 of file key.hpp.

```
00374 { return !(l > r); }
```

operator<= [8/9]

```
constexpr bool operator<= (
            std::int32_t l,
            Key r )  [friend]
```

Definition at line 373 of file key.hpp.

```
00373 { return !(l > r); }
```

operator<= [9/9]

```
constexpr bool operator<= (
            std::size_t l,
            Key r )  [friend]
```

Definition at line 375 of file key.hpp.

```
00375 { return !(l > r); }
```

operator== [1/9]

```
constexpr bool operator== (
            char l,
            Key r )  [friend]
```

Definition at line 319 of file key.hpp.

```
00319 { return Key(l) == r; }
```

operator== [2/9]

```
constexpr bool operator== (
            char32_t l,
            Key r )  [friend]
```

Definition at line 321 of file key.hpp.

```
00321 { return Key(l) == r; }
```

operator== [3/9]

```
constexpr bool operator== (
            Key l,
            char r )  [friend]
```

Definition at line 318 of file key.hpp.

```
00318 { return l == Key(r); }
```

operator== [4/9]

```
constexpr bool operator== (
            Key l,
            char32_t r )  [friend]
```

Definition at line 320 of file key.hpp.

```
00320 { return l == Key(r); }
```

operator== [5/9]

```
constexpr bool operator== (
            Key l,
            Key r )  [friend]
```

Definition at line 317 of file key.hpp.

```
00317 { return l.value == r.value; }
```

operator== [6/9]

```
constexpr bool operator== (
            Key l,
            std::int32_t r )  [friend]
```

Definition at line 322 of file key.hpp.

```
00322 { return l == Key(r); }
```

operator== [7/9]

```
constexpr bool operator== (
            Key l,
            std::size_t r )  [friend]
```

Definition at line 324 of file key.hpp.

```
00324 { return static_cast<std::size_t>(l.value) == r; }
```

operator== [8/9]

```
constexpr bool operator== (
            std::int32_t l,
            Key r )  [friend]
```

Definition at line 323 of file key.hpp.

```
00323 { return Key(l) == r; }
```

operator== [9/9]

```
constexpr bool operator== (
            std::size_t l,
            Key r )  [friend]
```

Definition at line 325 of file key.hpp.

```
00325 { return l == static_cast<std::size_t>(r.value); }
```

operator> [1/9]

```
constexpr bool operator> (
            char l,
            Key r )  [friend]
```

Definition at line 359 of file key.hpp.

```
00359 { return r < l; }
```

operator> [2/9]

```
constexpr bool operator> (
            char32_t l,
            Key r )  [friend]
```

Definition at line 361 of file key.hpp.

```
00361 { return r < l; }
```

operator> [3/9]

```
constexpr bool operator> (
            Key l,
            char r )  [friend]
```

Definition at line 358 of file key.hpp.

```
00358 { return r < l; }
```

operator> [4/9]

```
constexpr bool operator> (
            Key l,
            char32_t r )  [friend]
```

Definition at line 360 of file key.hpp.

```
00360 { return r < l; }
```

operator> [5/9]

```
constexpr bool operator> (
            Key l,
            Key r )  [friend]
```

Definition at line 357 of file key.hpp.

```
00357 { return r < l; }
```

operator> [6/9]

```
constexpr bool operator> (
            Key l,
            std::int32_t r )  [friend]
```

Definition at line 362 of file key.hpp.

```
00362 { return r < l; }
```

operator> [7/9]

```
constexpr bool operator> (
            Key l,
            std::size_t r )  [friend]
```

Definition at line 364 of file key.hpp.

```
00364 { return r < l; }
```

operator> [8/9]

```
constexpr bool operator> (
            std::int32_t l,
            Key r )  [friend]
```

Definition at line 363 of file key.hpp.

```
00363 { return r < l; }
```

operator> [9/9]

```
constexpr bool operator> (
            std::size_t l,
            Key r )  [friend]
```

Definition at line 365 of file key.hpp.

```
00365 { return r < l; }
```

operator>= [1/9]

```
constexpr bool operator>= (
            char l,
            Key r )  [friend]
```

Definition at line 349 of file key.hpp.

```
00349 { return !(l < r); }
```

operator>= [2/9]

```
constexpr bool operator>= (
            char32_t l,
            Key r )  [friend]
```

Definition at line 351 of file key.hpp.

```
00351 { return !(l < r); }
```

operator>= [3/9]

```
constexpr bool operator>= (
            Key l,
            char r )  [friend]
```

Definition at line 348 of file key.hpp.

```
00348 { return !(l < r); }
```

operator>= [4/9]

```
constexpr bool operator>= (
            Key l,
            char32_t r )  [friend]
```

Definition at line 350 of file key.hpp.

```
00350 { return !(l < r); }
```

operator>= [5/9]

```
constexpr bool operator>= (
            Key l,
            Key r )  [friend]
```

Definition at line 347 of file key.hpp.

```
00347 { return !(l < r); }
```

operator>= [6/9]

```
constexpr bool operator>= (
            Key l,
            std::int32_t r )  [friend]
```

Definition at line 352 of file key.hpp.

```
00352 { return !(l < r); }
```

operator>= [7/9]

```
constexpr bool operator>= (
            Key l,
            std::size_t r )  [friend]
```

Definition at line 354 of file key.hpp.

```
00354 { return !(l < r); }
```

operator>= [8/9]

```
constexpr bool operator>= (
            std::int32_t l,
            Key r )  [friend]
```

Definition at line 353 of file key.hpp.
```
00353 { return !(l < r); }
```

operator>= [9/9]

```
constexpr bool operator>= (
            std::size_t l,
            Key r )  [friend]
```

Definition at line 355 of file key.hpp.
```
00355 { return !(l < r); }
```

8.19.7 Member Data Documentation

value

```
std::int32_t Term::Key::value
```

Definition at line 421 of file key.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/key.hpp
- cpp-terminal/key.cpp

8.20 Term::MetaKey Class Reference

```
#include <cpp-terminal/key.hpp>
```

Public Types

- enum class Value : std::int32_t { None = 0 , Alt = (1UL << 22UL) , Ctrl = (1UL << 23UL) }

Public Member Functions

- constexpr MetaKey ()
- constexpr MetaKey (const MetaKey &key)=default
- MetaKey & operator= (const MetaKey &key)=default
- constexpr MetaKey (const Value &v)
- MetaKey & operator= (const Value &v)
- constexpr MetaKey (std::int32_t val)
- MetaKey & operator= (std::int32_t val)
- constexpr operator std::int32_t () const
- constexpr bool hasAlt () const
- constexpr bool hasCtrl () const
- MetaKey & operator+= (MetaKey r)
- MetaKey & operator+= (MetaKey::Value r)

Public Attributes

- std::int32_t value

Friends

- constexpr MetaKey operator+ (MetaKey l, MetaKey r)
- constexpr MetaKey operator+ (MetaKey::Value l, MetaKey::Value r)
- constexpr MetaKey operator+ (MetaKey l, MetaKey::Value r)
- constexpr MetaKey operator+ (MetaKey::Value l, MetaKey r)
- constexpr bool operator== (MetaKey l, MetaKey r)
- constexpr bool operator== (MetaKey l, MetaKey::Value r)
- constexpr bool operator== (MetaKey::Value l, MetaKey r)
- constexpr bool operator!= (MetaKey l, MetaKey r)
- constexpr bool operator!= (MetaKey l, MetaKey::Value r)
- constexpr bool operator!= (MetaKey::Value l, MetaKey r)
- constexpr bool operator< (MetaKey l, MetaKey r)
- constexpr bool operator>= (MetaKey l, MetaKey r)
- constexpr bool operator>= (MetaKey l, MetaKey::Value r)
- constexpr bool operator>= (MetaKey::Value l, MetaKey r)
- constexpr bool operator> (MetaKey l, MetaKey r)
- constexpr bool operator> (MetaKey l, MetaKey::Value r)
- constexpr bool operator> (MetaKey::Value l, MetaKey r)
- constexpr bool operator<= (MetaKey l, MetaKey r)
- constexpr bool operator<= (MetaKey l, MetaKey::Value r)
- constexpr bool operator<= (MetaKey::Value l, MetaKey r)

8.20.1 Detailed Description

Definition at line 18 of file key.hpp.

8.20.2 Member Enumeration Documentation

Value

```
enum class Term::MetaKey::Value :  std::int32_t  [strong]
```

Enumerator

None	
Alt	
Ctrl	

Definition at line 21 of file key.hpp.

```
00022   {
00023     // Last utf8 codepoint is U+10FFFF (00010000111111111111111111) So:
00024     None = 0,
00025     Alt  = (1UL << 22UL),
00026     Ctrl = (1UL << 23UL),
00027   };
```

8.20.3 Constructor & Destructor Documentation

MetaKey() [1/4]

```
constexpr Term::MetaKey::MetaKey ( )  [inline], [constexpr]
```

Definition at line 29 of file key.hpp.

```
00029 : value(static_cast<std::int32_t>(Value::None)) {}
```

MetaKey() [2/4]

```
constexpr Term::MetaKey::MetaKey (
            const MetaKey & key )  [constexpr], [default]
```

MetaKey() [3/4]

```
constexpr Term::MetaKey::MetaKey (
            const Value & v )  [inline], [constexpr]
```

Definition at line 33 of file key.hpp.

```
00033 : value(static_cast<std::int32_t>(v)) {}
```

MetaKey() [4/4]

```
constexpr Term::MetaKey::MetaKey (
            std::int32_t val )  [inline], [explicit], [constexpr]
```

Definition at line 40 of file key.hpp.

```
00040 : value(val) {}
```

8.20.4 Member Function Documentation

hasAlt()

```
constexpr bool Term::MetaKey::hasAlt ( ) const  [inline], [constexpr]
```

Definition at line 49 of file key.hpp.

```
00049 { return (this->value & static_cast<std::int32_t>(MetaKey::Value::Alt)) ==
      static_cast<std::int32_t>(MetaKey::Value::Alt); }
```

hasCtrl()

```
constexpr bool Term::MetaKey::hasCtrl ( ) const  [inline], [constexpr]
```

Definition at line 50 of file key.hpp.

```
00050 { return (this->value & static_cast<std::int32_t>(MetaKey::Value::Ctrl)) ==
      static_cast<std::int32_t>(MetaKey::Value::Ctrl); }
```

operator std::int32_t()

```
constexpr Term::MetaKey::operator std::int32_t ( ) const  [inline], [explicit], [constexpr]
```

Definition at line 47 of file key.hpp.

```
00047 { return this->value; }
```

operator+=() [1/2]

```
MetaKey & Term::MetaKey::operator+= (
            MetaKey r )  [inline]
```

Definition at line 57 of file key.hpp.

```
00057 { return *this = *this + r; }
```

operator+=() [2/2]

```
MetaKey & Term::MetaKey::operator+= (
            MetaKey::Value r )  [inline]
```

Definition at line 58 of file key.hpp.

```
00058 { return *this = *this + r; }
```

operator=() [1/3]

```
MetaKey & Term::MetaKey::operator= (
            const MetaKey & key )  [inline], [default]
```

operator=() [2/3]

```
MetaKey & Term::MetaKey::operator= (
            const Value & v )  [inline]
```

Definition at line 34 of file key.hpp.

```
00035   {
00036     this->value = static_cast<std::int32_t>(v);
00037     return *this;
00038   }
```

operator=() [3/3]

```
MetaKey & Term::MetaKey::operator= (
            std::int32_t val )  [inline]
```

Definition at line 41 of file key.hpp.

```
00042   {
00043     this->value = val;
00044     return *this;
00045   }
```

8.20.5 Friends And Related Symbol Documentation

operator"!= [1/3]

```
constexpr bool operator!= (
            MetaKey l,
            MetaKey r )  [friend]
```

Definition at line 64 of file key.hpp.

```
00064 { return !(l == r); }
```

operator"!= [2/3]

```
constexpr bool operator!= (
            MetaKey l,
            MetaKey::Value r )  [friend]
```

Definition at line 65 of file key.hpp.

```
00065 { return !(l == r); }
```

operator"!= [3/3]

```
constexpr bool operator!= (
            MetaKey::Value l,
            MetaKey r )  [friend]
```

Definition at line 66 of file key.hpp.

```
00066 { return !(l == r); }
```

operator+ [1/4]

```
constexpr MetaKey operator+ (
            MetaKey l,
            MetaKey r )  [friend]
```

Definition at line 52 of file key.hpp.

```
00052 { return MetaKey(l.value | r.value); }
```

operator+ [2/4]

```
constexpr MetaKey operator+ (
            MetaKey l,
            MetaKey::Value r )  [friend]
```

Definition at line 54 of file key.hpp.

```
00054 { return l + MetaKey(r); }
```

operator+ [3/4]

```
constexpr MetaKey operator+ (
            MetaKey::Value l,
            MetaKey r )  [friend]
```

Definition at line 55 of file key.hpp.

```
00055 { return MetaKey(l) + r; }
```

operator+ [4/4]

```
constexpr MetaKey operator+ (
            MetaKey::Value l,
            MetaKey::Value r )  [friend]
```

Definition at line 53 of file key.hpp.

```
00053 { return MetaKey(l) + MetaKey(r); }
```

operator<

```
constexpr bool operator< (
            MetaKey l,
            MetaKey r )  [friend]
```

Definition at line 68 of file key.hpp.

```
00068 { return l.value < r.value; }
```

operator<= [1/3]

```
constexpr bool operator<= (
            MetaKey l,
            MetaKey r )  [friend]
```

Definition at line 78 of file key.hpp.

```
00078 { return !(l > r); }
```

operator<= [2/3]

```
constexpr bool operator<= (
            MetaKey l,
            MetaKey::Value r )  [friend]
```

Definition at line 79 of file key.hpp.

```
00079 { return !(l > r); }
```

operator<= [3/3]

```
constexpr bool operator<= (
            MetaKey::Value l,
            MetaKey r )  [friend]
```

Definition at line 80 of file key.hpp.

```
00080 { return !(l > r); }
```

operator== [1/3]

```
constexpr bool operator== (
            MetaKey l,
            MetaKey r )  [friend]
```

Definition at line 60 of file key.hpp.
```
00060 { return l.value == r.value; }
```

operator== [2/3]

```
constexpr bool operator== (
            MetaKey l,
            MetaKey::Value r )  [friend]
```

Definition at line 61 of file key.hpp.
```
00061 { return l == MetaKey(r); }
```

operator== [3/3]

```
constexpr bool operator== (
            MetaKey::Value l,
            MetaKey r )  [friend]
```

Definition at line 62 of file key.hpp.
```
00062 { return MetaKey(l) == r; }
```

operator> [1/3]

```
constexpr bool operator> (
            MetaKey l,
            MetaKey r )  [friend]
```

Definition at line 74 of file key.hpp.
```
00074 { return r < l; }
```

operator> [2/3]

```
constexpr bool operator> (
            MetaKey l,
            MetaKey::Value r )  [friend]
```

Definition at line 75 of file key.hpp.
```
00075 { return r < l; }
```

operator> [3/3]

```
constexpr bool operator> (
            MetaKey::Value l,
            MetaKey r )  [friend]
```

Definition at line 76 of file key.hpp.
```
00076 { return r < l; }
```

operator>= [1/3]

```
constexpr bool operator>= (
            MetaKey l,
            MetaKey r )  [friend]
```

Definition at line 70 of file key.hpp.

```
00070 { return !(l < r); }
```

operator>= [2/3]

```
constexpr bool operator>= (
            MetaKey l,
            MetaKey::Value r )  [friend]
```

Definition at line 71 of file key.hpp.

```
00071 { return !(l < r); }
```

operator>= [3/3]

```
constexpr bool operator>= (
            MetaKey::Value l,
            MetaKey r )  [friend]
```

Definition at line 72 of file key.hpp.

```
00072 { return !(l < r); }
```

8.20.6 Member Data Documentation

value

```
std::int32_t Term::MetaKey::value
```

Definition at line 82 of file key.hpp.

The documentation for this class was generated from the following file:

- cpp-terminal/key.hpp

8.21 Term::Model Class Reference

```
#include <cpp-terminal/prompt.hpp>
```

Public Attributes

- std::string prompt_string
- std::vector< std::string > lines {""}
- std::size_t cursor_col {1}
- std::size_t cursor_row {1}

8.21.1 Detailed Description

Definition at line 65 of file prompt.hpp.

8.21.2 Member Data Documentation

cursor_col

```
std::size_t Term::Model::cursor_col {1}
```

Definition at line 71 of file prompt.hpp.
```
00071 {1};
```

cursor_row

```
std::size_t Term::Model::cursor_row {1}
```

Definition at line 72 of file prompt.hpp.
```
00072 {1};
```

lines

```
std::vector<std::string> Term::Model::lines {""}
```

Definition at line 69 of file prompt.hpp.
```
00069 {""};      // The current input string in the prompt as a vector of lines, without '\n' at the end.
```

prompt_string

```
std::string Term::Model::prompt_string
```

Definition at line 68 of file prompt.hpp.

The documentation for this class was generated from the following file:

- cpp-terminal/prompt.hpp

8.22 Term::Mouse Class Reference

```
#include <cpp-terminal/mouse.hpp>
```

Public Member Functions

- Mouse ()=default
- Mouse (const Term::Button &buttons, const std::uint16_t &row, const std::uint16_t &column)
- std::size_t row () const noexcept
- std::size_t column () const noexcept
- Term::Button getButton () const noexcept
- bool is (const Term::Button::Type &type, const Term::Button::Action &action) const noexcept
- bool is (const Term::Button::Type &type) const noexcept
- bool operator== (const Term::Mouse &mouse) const
- bool operator!= (const Term::Mouse &mouse) const

Private Attributes

- Term::Button m_buttons
- std::uint16_t m_row {0}
- std::uint16_t m_column {0}

8.22.1 Detailed Description

Definition at line 59 of file mouse.hpp.

8.22.2 Constructor & Destructor Documentation

Mouse() [1/2]

```
Term::Mouse::Mouse ( )  [default]
```

Mouse() [2/2]

```
Term::Mouse::Mouse (
            const Term::Button & buttons,
            const std::uint16_t & row,
            const std::uint16_t & column )  [inline]
```

Definition at line 63 of file mouse.hpp.

```
00063 : m_buttons(buttons), m_row(row), m_column(column) {}
```

8.22.3 Member Function Documentation

column()

```
std::size_t Term::Mouse::column ( ) const  [noexcept]
```

Definition at line 28 of file mouse.cpp.

```
00028 { return static_cast<std::size_t>(m_column); }
```

getButton()

```
Term::Button Term::Mouse::getButton ( ) const  [noexcept]
```

Definition at line 24 of file mouse.cpp.

```
00024 { return m_buttons; }
```

is() [1/2]

```
bool Term::Mouse::is (
            const Term::Button::Type & type ) const  [noexcept]
```

Definition at line 22 of file mouse.cpp.

```
00022 { return m_buttons.type() == type; }
```

is() [2/2]

```
bool Term::Mouse::is (
            const Term::Button::Type & type,
            const Term::Button::Action & action ) const  [noexcept]
```

Definition at line 20 of file mouse.cpp.

```
00020 { return (m_buttons.type() == type) && (m_buttons.action() == action); }
```

operator"!=()

```
bool Term::Mouse::operator!= (
            const Term::Mouse & mouse ) const
```

Definition at line 31 of file mouse.cpp.

```
00031 { return !(*this == mouse); }
```

operator==()

```
bool Term::Mouse::operator== (
            const Term::Mouse & mouse ) const
```

Definition at line 30 of file mouse.cpp.

```
00030 { return (m_row == mouse.m_row) && (m_column == mouse.m_column) && (m_buttons == mouse.m_buttons); }
```

row()

```
std::size_t Term::Mouse::row ( ) const  [noexcept]
```

Definition at line 26 of file mouse.cpp.

```
00026 { return static_cast<std::size_t>(m_row); }
```

8.22.4 Member Data Documentation

m_buttons

```
Term::Button Term::Mouse::m_buttons  [private]
```

Definition at line 73 of file mouse.hpp.

m_column

```
std::uint16_t Term::Mouse::m_column {0}  [private]
```

Definition at line 75 of file mouse.hpp.

```
00075 {0};
```

m_row

```
std::uint16_t Term::Mouse::m_row {0}  [private]
```

Definition at line 74 of file mouse.hpp.
```
00074 {0};
```

The documentation for this class was generated from the following files:

- cpp-terminal/mouse.hpp
- cpp-terminal/mouse.cpp

8.23 Term::Options Class Reference

```
#include <cpp-terminal/options.hpp>
```

Public Member Functions

- Options ()=default
- Options (const std::initializer_list< Term::Option > &option)
- template<typename... Args>
  Options (const Args &&... args)
- bool operator== (const Options &options)
- bool operator!= (const Options &options)
- bool has (const Option &option) const noexcept

Private Member Functions

- void clean ()

Private Attributes

- std::vector< Term::Option > m_Options

8.23.1 Detailed Description

Definition at line 34 of file options.hpp.

8.23.2 Constructor & Destructor Documentation

Options() [1/3]

```
Term::Options::Options ( )  [default]
```

Options() [2/3]

```
Term::Options::Options (
            const std::initializer_list< Term::Option > & option )
```

Definition at line 14 of file options.cpp.

```
00014 : m_Options(option) { clean(); }
```

Options() [3/3]

```
template<typename... Args>
Term::Options::Options (
            const Args &&... args )  [inline], [explicit]
```

Definition at line 39 of file options.hpp.

```
00039 : m_Options(std::initializer_list<Term::Option>{args...}) { clean(); }
```

8.23.3 Member Function Documentation

clean()

```
void Term::Options::clean ( )  [private]
```

Definition at line 19 of file options.cpp.

```
00020 {
00021   std::vector<Term::Option> cleaned;
00022   std::sort(m_Options.begin(), m_Options.end());
00023   while(!m_Options.empty())
00024   {
00025     const std::size_t count      = std::count(m_Options.begin(), m_Options.end(), m_Options[0]);
00026     const std::size_t anti_count = std::count(m_Options.begin(), m_Options.end(),
      static_cast<Term::Option>(-1 * static_cast<std::int16_t>(m_Options[0])));
00027     if(count > anti_count) { cleaned.emplace_back(m_Options[0]); }
00028     else if(count < anti_count) { cleaned.emplace_back(static_cast<Term::Option>(-1 *
      static_cast<std::int16_t>(m_Options[0]))); }
00029     m_Options.erase(std::remove(m_Options.begin(), m_Options.end(), static_cast<Term::Option>(-1 *
      static_cast<std::int16_t>(m_Options[0]))), m_Options.end());
00030     m_Options.erase(std::remove(m_Options.begin(), m_Options.end(), m_Options[0]), m_Options.end());
00031   }
00032   m_Options = cleaned;
00033 }
```

has()

```
bool Term::Options::has (
            const Option & option ) const  [noexcept]
```

Definition at line 35 of file options.cpp.

```
00035 { return std::find(m_Options.begin(), m_Options.end(), option) != m_Options.end(); }
```

operator"!=()

```
bool Term::Options::operator!= (
            const Options & options )
```

Definition at line 17 of file options.cpp.

```
00017 { return !(m_Options == options.m_Options); }
```

operator==()

```
bool Term::Options::operator== (
            const Options & options )
```

Definition at line 16 of file options.cpp.

```
00016 { return m_Options == options.m_Options; }
```

8.23.4 Member Data Documentation

m_Options

```
std::vector<Term::Option> Term::Options::m_Options  [private]
```

Definition at line 47 of file options.hpp.

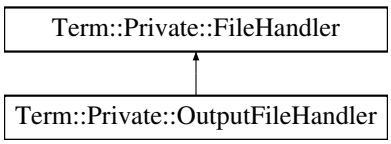
The documentation for this class was generated from the following files:

- cpp-terminal/options.hpp
- cpp-terminal/options.cpp

8.24 Term::Private::OutputFileHandler Class Reference

```
#include <cpp-terminal/private/file.hpp>
```

Inheritance diagram for Term::Private::OutputFileHandler:



Public Member Functions

- OutputFileHandler (std::recursive_mutex &io_mutex) noexcept
- OutputFileHandler (const OutputFileHandler &other)=delete
- OutputFileHandler (OutputFileHandler &&other)=delete
- OutputFileHandler & operator= (OutputFileHandler &&rhs)=delete
- OutputFileHandler & operator= (const OutputFileHandler &rhs)=delete
- ~OutputFileHandler () override=default
- std::size_t write (const std::string &str) const
- std::size_t write (const char &character) const

Public Member Functions inherited from Term::Private::FileHandler

- FileHandler (std::recursive_mutex &mutex, const std::string &file, const std::string &mode) noexcept
- FileHandler (const FileHandler &)=delete
- FileHandler (FileHandler &&)=delete
- FileHandler & operator= (const FileHandler &)=delete
- FileHandler & operator= (FileHandler &&)=delete
- virtual ~FileHandler () noexcept
- Handle handle () const noexcept
- bool null () const noexcept
- std::FILE * file () const noexcept
- std::int32_t fd () const noexcept
- void lockIO ()
- void unlockIO ()
- void flush ()

Static Public Attributes

- static const constexpr char * m_file {"CONOUT$"}

Additional Inherited Members

Public Types inherited from Term::Private::FileHandler

- using Handle = void*

8.24.1 Detailed Description

Definition at line 56 of file file.hpp.

8.24.2 Constructor & Destructor Documentation

OutputFileHandler() [1/3]

```
Term::Private::OutputFileHandler::OutputFileHandler (
            std::recursive_mutex & io_mutex )  [explicit], [noexcept]
```

Definition at line 147 of file file.cpp.

```
00148     : FileHandler(io_mutex, m_file, "w")
00149 {
00150   //noop
00151 }
00152 catch(...)
00153 {
00154   ExceptionHandler(ExceptionDestination::StdErr);
00155 }
```

OutputFileHandler() [2/3]

```
Term::Private::OutputFileHandler::OutputFileHandler (
            const OutputFileHandler & other )  [delete]
```

OutputFileHandler() [3/3]

```
Term::Private::OutputFileHandler::OutputFileHandler (
            OutputFileHandler && other )  [delete]
```

~OutputFileHandler()

```
Term::Private::OutputFileHandler::~OutputFileHandler ( )  [override], [default]
```

8.24.3 Member Function Documentation

operator=() [1/2]

```
OutputFileHandler & Term::Private::OutputFileHandler::operator= (
            const OutputFileHandler & rhs )  [delete]
```

operator=() [2/2]

```
OutputFileHandler & Term::Private::OutputFileHandler::operator= (
            OutputFileHandler && rhs )  [delete]
```

write() [1/2]

```
std::size_t Term::Private::OutputFileHandler::write (
            const char & character ) const
```

Definition at line 109 of file file.cpp.

```
00110 {
00111 #if defined(_WIN32)
00112   DWORD written{0};
00113   Term::Private::WindowsError().check_if(0 == WriteConsole(handle(), &character, 1, &written,
      nullptr)).throw_exception("WriteConsole(handle(), &character, 1, &written, nullptr)");
00114   return static_cast<std::size_t>(written);
00115 #else
00116   ssize_t written{0};
00117   Term::Private::Errno().check_if((written = ::write(fd(), &character, 1)) ==
      -1).throw_exception("::write(fd(), &character, 1)");
00118   return static_cast<std::size_t>(written);
00119 #endif
00120 }
```

write() [2/2]

```
std::size_t Term::Private::OutputFileHandler::write (
            const std::string & str ) const
```

Definition at line 96 of file file.cpp.

```
00097 {
00098 #if defined(_WIN32)
00099   DWORD written{0};
00100   if(!str.empty()) { Term::Private::WindowsError().check_if(0 == WriteConsole(handle(), &str[0],
      static_cast<DWORD>(str.size()), &written, nullptr)).throw_exception("WriteConsole(handle(), &str[0],
      static_cast<DWORD>(str.size()), &written, nullptr)"); }
00101   return static_cast<std::size_t>(written);
00102 #else
00103   ssize_t written{0};
00104   if(!str.empty()) { Term::Private::Errno().check_if((written = ::write(fd(), str.data(), str.size()))
      == -1).throw_exception("::write(fd(), str.data(), str.size())"); }
00105   return static_cast<std::size_t>(written);
00106 #endif
00107 }
```

8.24.4 Member Data Documentation

m_file

```
const constexpr char* Term::Private::OutputFileHandler::m_file {"CONOUT$"}  [static], [constexpr]
```

Definition at line 69 of file file.hpp.
```
00069 {"CONOUT$"};
```

The documentation for this class was generated from the following files:

- cpp-terminal/private/file.hpp
- cpp-terminal/private/file.cpp

8.25 Term::Screen Class Reference

```
#include <cpp-terminal/screen.hpp>
```

Public Member Functions

- Screen ()=default
- Screen (const std::size_t &rows, const std::size_t &columns)
- std::size_t rows () const
- std::size_t columns () const
- bool empty () const
- bool operator== (const Term::Screen &screen) const
- bool operator!= (const Term::Screen &screen) const

Private Attributes

- std::pair< std::size_t, std::size_t > m_size

8.25.1 Detailed Description

Definition at line 19 of file screen.hpp.

8.25.2 Constructor & Destructor Documentation

Screen() [1/2]

```
Term::Screen::Screen ( )  [default]
```

Screen() [2/2]

```
Term::Screen::Screen (
            const std::size_t & rows,
            const std::size_t & columns )
```

Definition at line 12 of file screen.cpp.
```
00012 : m_size({rows, columns}) {}
```

8.25.3 Member Function Documentation

columns()

```
std::size_t Term::Screen::columns ( ) const
```

Definition at line 16 of file screen.cpp.

```
00016 { return m_size.second; }
```

empty()

```
bool Term::Screen::empty ( ) const
```

Definition at line 18 of file screen.cpp.

```
00018 { return (0 == m_size.second) && (0 == m_size.first); }
```

operator"!=()

```
bool Term::Screen::operator!= (
            const Term::Screen & screen ) const
```

Definition at line 34 of file screen.cpp.

```
00034 { return !(*this == screen); }
```

operator==()

```
bool Term::Screen::operator== (
            const Term::Screen & screen ) const
```

Definition at line 32 of file screen.cpp.

```
00032 { return (this->rows() == screen.rows()) && (this->columns() == screen.columns()); }
```

rows()

```
std::size_t Term::Screen::rows ( ) const
```

Definition at line 14 of file screen.cpp.

```
00014 { return m_size.first; }
```

8.25.4 Member Data Documentation

m_size

```
std::pair<std::size_t, std::size_t> Term::Screen::m_size  [private]
```

Definition at line 31 of file screen.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/screen.hpp
- cpp-terminal/screen.cpp

8.26 Term::Private::Sigwinch Class Reference

`#include <cpp-terminal/private/sigwinch.hpp>`

Static Public Member Functions

- static void registerSigwinch ()
- static void blockSigwinch ()
- static void unblockSigwinch ()
- static bool isSigwinch (const std::int32_t &file_descriptor=-1) noexcept
- static std::int32_t get () noexcept

Static Private Attributes

- static std::int32_t m_fd {-1}

8.26.1 Detailed Description

Definition at line 19 of file sigwinch.hpp.

8.26.2 Member Function Documentation

blockSigwinch()

`void Term::Private::Sigwinch::blockSigwinch ( )  [static]`

Definition at line 65 of file sigwinch.cpp.

```
00066 {
00067 #if !defined(_WIN32)
00068   ::sigset_t windows_event = {};
00069   Term::Private::Errno().check_if(sigemptyset(&windows_event) !=
      0).throw_exception("sigemptyset(&windows_event)");
00070   Term::Private::Errno().check_if(sigaddset(&windows_event, SIGWINCH) !=
      0).throw_exception("sigaddset(&windows_event, SIGWINCH)");
00071   Term::Private::Errno().check_if(::pthread_sigmask(SIG_BLOCK, &windows_event, nullptr) !=
      0).throw_exception("::pthread_sigmask(SIG_BLOCK, &windows_event, nullptr)");
00072 #endif
00073 }
```

get()

`std::int32_t Term::Private::Sigwinch::get ( )  [static], [noexcept]`

Definition at line 38 of file sigwinch.cpp.

```
00039 {
00040 #if defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00041   return Term::Private::m_signalStatus;
00042 #else
00043   return m_fd;
00044 #endif
00045 }
```

isSigwinch()

```
bool Term::Private::Sigwinch::isSigwinch (
            const std::int32_t & file_descriptor = -1 )  [static], [noexcept]
```

Definition at line 85 of file sigwinch.cpp.

```
00086 {
00087 #if defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00088   if(Term::Private::m_signalStatus == 1)
00089   {
00090     static_cast<void>(file_descriptor);  // suppress warning
00091     Term::Private::m_signalStatus = {0};
00092     return true;
00093   }
00094   return false;
00095 #elif defined(__linux__)
00096   if(m_fd == file_descriptor)
00097   {
00098     // read it to clean
00099     ::signalfd_siginfo fdsi = {};
00100     ::read(m_fd, &fdsi, sizeof(fdsi));
00101     return true;
00102   }
00103   return false;
00104 #else
00105   static_cast<void>(file_descriptor);  // suppress warning
00106   return false;
00107 #endif
00108 }
```

registerSigwinch()

```
void Term::Private::Sigwinch::registerSigwinch ( )  [static]
```

Definition at line 49 of file sigwinch.cpp.

```
00050 {
00051 #if defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00052   struct sigaction sa;
00053   Term::Private::Errno().check_if(sigemptyset(&sa.sa_mask) !=
      0).throw_exception("sigemptyset(&sa.sa_mask)");
00054   sa.sa_flags   = {0};
00055   sa.sa_handler = {Term::Private::sigwinchHandler};
00056   Term::Private::Errno().check_if(sigaction(SIGWINCH, &sa, nullptr) !=
      0).throw_exception("sigaction(SIGWINCH, &sa, nullptr)");
00057 #elif defined(__linux__)
00058   ::sigset_t windows_event = {};
00059   Term::Private::Errno().check_if(sigemptyset(&windows_event) !=
      0).throw_exception("sigemptyset(&windows_event)");
00060   Term::Private::Errno().check_if(sigaddset(&windows_event, SIGWINCH) !=
      0).throw_exception("sigaddset(&windows_event, SIGWINCH)");
00061   Term::Private::Errno().check_if((m_fd = ::signalfd(-1, &windows_event, SFD_NONBLOCK | SFD_CLOEXEC))
      == -1).throw_exception("m_fd = ::signalfd(-1, &windows_event, SFD_NONBLOCK | SFD_CLOEXEC)");
00062 #endif
00063 }
```

unblockSigwinch()

```
void Term::Private::Sigwinch::unblockSigwinch ( )  [static]
```

Definition at line 75 of file sigwinch.cpp.

```
00076 {
00077 #if !defined(_WIN32)
00078   ::sigset_t windows_event = {};
00079   Term::Private::Errno().check_if(sigemptyset(&windows_event) !=
      0).throw_exception("sigemptyset(&windows_event)");
00080   Term::Private::Errno().check_if(sigaddset(&windows_event, SIGWINCH) !=
      0).throw_exception("sigaddset(&windows_event, SIGWINCH)");
00081   Term::Private::Errno().check_if(::pthread_sigmask(SIG_UNBLOCK, &windows_event, nullptr) !=
      0).throw_exception("::pthread_sigmask(SIG_UNBLOCK, &windows_event, nullptr)");
00082 #endif
00083 }
```

8.26.3 Member Data Documentation

m_fd

```
std::int32_t Term::Private::Sigwinch::m_fd {-1}  [static], [private]
```

Definition at line 47 of file sigwinch.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/private/sigwinch.hpp
- cpp-terminal/private/sigwinch.cpp

8.27 Term::Terminal Class Reference

```
#include <cpp-terminal/terminal_impl.hpp>
```

Public Member Functions

- ~Terminal () noexcept
- Terminal () noexcept
- Terminal (const Terminal &)=delete
- Terminal (Terminal &&)=delete
- Terminal & operator= (Terminal &&)=delete
- Terminal & operator= (const Terminal &)=delete
- template<typename... Args>
  void setOptions (const Args &&... args)
- Term::Options getOptions () const noexcept

Private Member Functions

- void setMode () const
  Set mode raw/cooked.
- void setOptions ()
- void applyOptions () const

Static Private Member Functions

- static void store_and_restore () noexcept
  Store and restore the default state of the terminal.
- static std::size_t setMouseEvents ()
- static std::size_t unsetMouseEvents ()
- static std::size_t setFocusEvents ()
- static std::size_t unsetFocusEvents ()
- static void set_unset_utf8 ()

Private Attributes

- Term::Options m_options

8.27.1 Detailed Description

Definition at line 20 of file terminal_impl.hpp.

8.27.2 Constructor & Destructor Documentation

~Terminal()

```
Term::Terminal::~Terminal ( )  [noexcept]
```

Definition at line 37 of file terminal_impl.cpp.

```
00039 {
00040   if(getOptions().has(Option::ClearScreen)) { Term::Private::out.write(clear_buffer() +
      style(Style::Reset) + cursor_move(1, 1) + screen_load()); }
00041   if(getOptions().has(Option::NoCursor)) { Term::Private::out.write(cursor_on()); }
00042   set_unset_utf8();
00043   store_and_restore();
00044   unsetFocusEvents();
00045   unsetMouseEvents();
00046 }
00047 catch(...)
00048 {
00049   ExceptionHandler(Private::ExceptionDestination::StdErr);
00050 }
```

Terminal() [1/3]

```
Term::Terminal::Terminal ( )  [noexcept]
```

Definition at line 23 of file terminal_impl.cpp.

```
00025 {
00026   Term::Private::Sigwinch::blockSigwinch();
00027   Term::Private::Sigwinch::registerSigwinch();
00028   store_and_restore();
00029   setMode();  //Save the default cpp-terminal mode done in store_and_restore();
00030   set_unset_utf8();
00031 }
00032 catch(...)
00033 {
00034   ExceptionHandler(Private::ExceptionDestination::StdErr);
00035 }
```

Terminal() [2/3]

```
Term::Terminal::Terminal (
            const Terminal &  )  [delete]
```

Terminal() [3/3]

```
Term::Terminal::Terminal (
            Terminal &&  )  [delete]
```

8.27.3 Member Function Documentation

applyOptions()

```
void Term::Terminal::applyOptions ( ) const  [private]
```

Definition at line 52 of file terminal_impl.cpp.

```
00053 {
00054   if(getOptions().has(Option::ClearScreen)) { Term::Private::out.write(screen_save() + clear_buffer()
      + style(Style::Reset) + cursor_move(1, 1)); }
00055   if(getOptions().has(Option::NoCursor)) { Term::Private::out.write(cursor_off()); }
00056   setMode();
00057 }
```

getOptions()

```
Term::Options Term::Terminal::getOptions ( ) const  [noexcept]
```

Definition at line 21 of file terminal_impl.cpp.

```
00021 { return m_options; }
```

operator=() [1/2]

```
Terminal & Term::Terminal::operator= (
            const Terminal &  )  [delete]
```

operator=() [2/2]

```
Terminal & Term::Terminal::operator= (
            Terminal &&  )  [delete]
```

set_unset_utf8()

```
void Term::Terminal::set_unset_utf8 ( )  [static], [private]
```

Definition at line 32 of file terminal_impl.cpp.

```
00033 {
00034   static bool enabled{false};
00035 #if defined(_WIN32)
00036   static UINT out_code_page{0};
00037   static UINT in_code_page{0};
00038   if(!enabled)
00039   {
00040     if((out_code_page = GetConsoleOutputCP()) == 0) throw
      Term::Private::WindowsException(GetLastError(), "GetConsoleOutputCP()");
00041     if(!SetConsoleOutputCP(CP_UTF8)) throw Term::Private::WindowsException(GetLastError(),
      "SetConsoleOutputCP(CP_UTF8)");
00042     if((in_code_page = GetConsoleCP()) == 0) throw Term::Private::WindowsException(GetLastError(),
      "GetConsoleCP()");
00043     if(!SetConsoleCP(CP_UTF8)) throw Term::Private::WindowsException(GetLastError(),
      "SetConsoleCP(CP_UTF8)");
00044     enabled = true;
00045   }
00046   else
00047   {
00048     if(!SetConsoleOutputCP(out_code_page)) throw Term::Private::WindowsException(GetLastError(),
      "SetConsoleOutputCP(out_code_page)");
00049     if(!SetConsoleCP(in_code_page)) throw Term::Private::WindowsException(GetLastError(),
      "SetConsoleCP(in_code_page)");
00050   }
00051 #else
```

```
00052   if(!enabled)
00053   {
00054     const Term::Cursor cursor_before{Term::cursor_position()};
00055     Term::Private::out.write("\u001b%G");  // Try to activate UTF-8 (NOT warranty)
00056     const std::string  read{Term::Private::in.read()};
00057     const Term::Cursor cursor_after{Term::cursor_position()};
00058     const std::size_t  moved{cursor_after.column() - cursor_before.column()};
00059     std::string        rem;
00060     rem.reserve(moved * 3);
00061     for(std::size_t i = 0; i != moved; ++i) { rem += "\b \b"; }
00062     Term::Private::out.write(rem);
00063     enabled = 0 == moved;
00064   }
00065   else
00066   {
00067     // Does not return the original charset but, the default defined by standard ISO 8859-1 (ISO
      2022);
00068     Term::Private::out.write("\u001b%@");
00069   }
00070 #endif
00071 }
```

setFocusEvents()

```
std::size_t Term::Terminal::setFocusEvents ( )  [static], [private]
```

Definition at line 142 of file terminal_impl.cpp.

```
00143 {
00144 #if defined(_WIN32)
00145   return static_cast<std::size_t>(ENABLE_WINDOW_INPUT);
00146 #else
00147   return Term::Private::out.write("\u001b[?1004h");
00148 #endif
00149 }
```

setMode()

```
void Term::Terminal::setMode ( ) const  [private]
```

Set mode raw/cooked.

First call is to save the good state set-up by cpp-terminal.

Definition at line 160 of file terminal_impl.cpp.

```
00161 {
00162   static bool activated{false};
00163 #if defined(_WIN32)
00164   static DWORD flags{0};
00165   if(!activated)
00166   {
00167     if(!Private::out.null())
00168       if(!GetConsoleMode(Private::in.handle(), &flags)) { throw
      Term::Private::WindowsException(GetLastError()); }
00169     activated = true;
00170     return;
00171   }
00172   DWORD send = flags;
00173   if(m_options.has(Option::Raw))
00174   {
00175     send &= ~(ENABLE_LINE_INPUT | ENABLE_ECHO_INPUT | ENABLE_PROCESSED_INPUT);
00176     send |= (setFocusEvents() | setMouseEvents());
00177   }
00178   else if(m_options.has(Option::Cooked))
00179   {
00180     send |= (ENABLE_LINE_INPUT | ENABLE_ECHO_INPUT | ENABLE_PROCESSED_INPUT);
00181     send &= ~(setFocusEvents() | setMouseEvents());
00182   }
00183   if(m_options.has(Option::NoSignalKeys)) { send &= ~ENABLE_PROCESSED_INPUT; }
00184   else if(m_options.has(Option::SignalKeys)) { send |= ENABLE_PROCESSED_INPUT; }
00185   if(!Private::out.null())
00186     if(!SetConsoleMode(Private::in.handle(), send)) { throw
      Term::Private::WindowsException(GetLastError()); }
00187 #else
00188   if(!Private::out.null())
```

```
00189   {
00190     static ::termios raw = {};
00191     if(!activated)
00192     {
00193       Term::Private::Errno().check_if(tcgetattr(Private::out.fd(), &raw) ==
      -1).throw_exception("tcgetattr(Private::out.fd(), &raw)");
00194       raw.c_cflag &= ~static_cast<std::size_t>(CSIZE | PARENB);
00195       raw.c_cflag |= CS8;
00196       raw.c_cc[VMIN]  = 1;
00197       raw.c_cc[VTIME] = 0;
00198       activated       = true;
00199       return;
00200     }
00201     ::termios send = raw;
00202     if(m_options.has(Option::Raw))
00203     {
00204       send.c_iflag &= ~static_cast<std::size_t>(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR |
      ICRNL | IXON | INPCK);
00205       // This disables output post-processing, requiring explicit \r\n. We
00206       // keep it enabled, so that in C++, one can still just use std::endl
00207       // for EOL instead of "\r\n".
00208       //send.c_oflag &= ~static_cast<std::size_t>(OPOST);
00209       send.c_lflag &= ~static_cast<std::size_t>(ECHO | ECHONL | ICANON | ISIG | IEXTEN);
00210       setMouseEvents();
00211       setFocusEvents();
00212     }
00213     else if(m_options.has(Option::Cooked))
00214     {
00215       send = raw;
00216       unsetMouseEvents();
00217       unsetFocusEvents();
00218     }
00219     if(m_options.has(Option::NoSignalKeys)) { send.c_lflag &= ~static_cast<std::size_t>(ISIG); }
      //FIXME need others flags !
00220     else if(m_options.has(Option::NoSignalKeys)) { send.c_lflag |= ISIG; }
00221     Term::Private::Errno().check_if(tcsetattr(Private::out.fd(), TCSAFLUSH, &send) ==
      -1).throw_exception("tcsetattr(Private::out.fd(), TCSAFLUSH, &send)");
00222   }
00223 #endif
00224 }
```

setMouseEvents()

```
std::size_t Term::Terminal::setMouseEvents ( )  [static], [private]
```

Definition at line 124 of file terminal_impl.cpp.

```
00125 {
00126 #if defined(_WIN32)
00127   return static_cast<std::size_t>(ENABLE_MOUSE_INPUT);
00128 #else
00129   return Term::Private::out.write("\u001b[?1002h\u001b[?1003h\u001b[?1006h");
00130 #endif
00131 }
```

setOptions() [1/2]

```
void Term::Terminal::setOptions ( )  [private]
```

setOptions() [2/2]

```
template<typename...  Args>
void Term::Terminal::setOptions (
            const Args &&...  args )  [inline]
```

Definition at line 29 of file terminal_impl.hpp.

```
00030   {
00031     m_options = {args...};
00032     applyOptions();
00033   }
```

store_and_restore()

```
void Term::Terminal::store_and_restore ( )  [static], [private], [noexcept]
```

Store and restore the default state of the terminal.

Configure the default mode for cpp-terminal.

Definition at line 73 of file terminal_impl.cpp.

```
00075 {
00076   static bool enabled{false};
00077 #if defined(_WIN32)
00078   static DWORD originalOut{0};
00079   static DWORD originalIn{0};
00080   if(!enabled)
00081   {
00082     Term::Private::WindowsError().check_if(GetConsoleMode(Private::out.handle(), &originalOut) ==
      0).throw_exception("GetConsoleMode(Private::out.handle(), &originalOut)");
00083     Term::Private::WindowsError().check_if(GetConsoleMode(Private::in.handle(), &originalIn) ==
      0).throw_exception("GetConsoleMode(Private::in.handle(), &originalIn)");
00084     DWORD in{static_cast<DWORD>((originalIn & ~(ENABLE_QUICK_EDIT_MODE | setFocusEvents() |
      setMouseEvents())) | (ENABLE_EXTENDED_FLAGS))};
00085     DWORD out{originalOut};
00086     // Check if ENABLE_VIRTUAL_TERMINAL_PROCESSING | DISABLE_NEWLINE_AUTO_RETURN can be activated, if
      not we are a legacy terminal.
00087     DWORD test = out;
00088     test |= ENABLE_VIRTUAL_TERMINAL_PROCESSING;
00089     if(!SetConsoleMode(Private::out.handle(), test)) { SetConsoleMode(Private::out.handle(), out); }
00090     else
00091     {
00092       out |= ENABLE_VIRTUAL_TERMINAL_PROCESSING | DISABLE_NEWLINE_AUTO_RETURN;
00093       in |= ENABLE_VIRTUAL_TERMINAL_INPUT;
00094     }
00095     if(!SetConsoleMode(Private::out.handle(), out)) { throw
      Term::Private::WindowsException(GetLastError(), "SetConsoleMode(Private::out.handle()"); }
00096     if(!SetConsoleMode(Private::in.handle(), in)) { throw
      Term::Private::WindowsException(GetLastError(), "SetConsoleMode(Private::in.handle(), in)"); }
00097     enabled = true;
00098   }
00099   else
00100   {
00101     if(!SetConsoleMode(Private::out.handle(), originalOut)) { throw
      Term::Private::WindowsException(GetLastError(), "SetConsoleMode(Private::out.handle(), originalOut)");
      }
00102     if(!SetConsoleMode(Private::in.handle(), originalIn)) { throw
      Term::Private::WindowsException(GetLastError(), "SetConsoleMode(Private::in.handle(), originalIn)"); }
00103   }
00104 #else
00105   static termios orig_termios;
00106   if(!enabled)
00107   {
00108     if(!Private::out.null()) { Term::Private::Errno().check_if(tcgetattr(Private::out.fd(),
      &orig_termios) == -1).throw_exception("tcgetattr() failed"); }
00109     enabled = true;
00110   }
00111   else
00112   {
00113     unsetMouseEvents();
00114     unsetFocusEvents();
00115     if(!Private::out.null()) { Term::Private::Errno().check_if(tcsetattr(Private::out.fd(), TCSAFLUSH,
      &orig_termios) == -1).throw_exception("tcsetattr() failed in destructor"); }
00116   }
00117 #endif
00118 }
00119 catch(...)
00120 {
00121   ExceptionHandler(Private::ExceptionDestination::StdErr);
00122 }
```

unsetFocusEvents()

```
std::size_t Term::Terminal::unsetFocusEvents ( )  [static], [private]
```

Definition at line 151 of file terminal_impl.cpp.

```
00152 {
00153 #if defined(_WIN32)
00154   return static_cast<std::size_t>(ENABLE_WINDOW_INPUT);
00155 #else
00156   return Term::Private::out.write("\u001b[?1004l");
00157 #endif
00158 }
```

unsetMouseEvents()

```
std::size_t Term::Terminal::unsetMouseEvents ( )  [static], [private]
```

Definition at line 133 of file terminal_impl.cpp.

```
00134 {
00135 #if defined(_WIN32)
00136   return static_cast<std::size_t>(ENABLE_MOUSE_INPUT);
00137 #else
00138   return Term::Private::out.write("\u001b[?1006l\u001b[?1003l\u001b[?1002l");
00139 #endif
00140 }
```

8.27.4 Member Data Documentation

m_options

```
Term::Options Term::Terminal::m_options  [private]
```

Definition at line 57 of file terminal_impl.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/terminal_impl.hpp
- cpp-terminal/private/terminal_impl.cpp
- cpp-terminal/terminal_impl.cpp

8.28 Term::TerminalInitializer Class Reference

```
#include <cpp-terminal/terminal_initializer.hpp>
```

Public Member Functions

- ~TerminalInitializer () noexcept
- TerminalInitializer () noexcept
- TerminalInitializer (const TerminalInitializer &)=delete
- TerminalInitializer (TerminalInitializer &&)=delete
- TerminalInitializer & operator= (TerminalInitializer &&)=delete
- TerminalInitializer & operator= (const TerminalInitializer &)=delete

Static Private Attributes

- static std::size_t m_counter {0}

8.28.1 Detailed Description

Definition at line 17 of file terminal_initializer.hpp.

8.28.2 Constructor & Destructor Documentation

~TerminalInitializer()

```
Term::TerminalInitializer::~TerminalInitializer ( )  [noexcept]
```

Definition at line 35 of file terminal_initializer.cpp.

```
00037 {
00038   --m_counter;
00039   if(0 == m_counter) { (&Term::terminal)->~Terminal(); }
00040 }
00041 catch(...)
00042 {
00043   ExceptionHandler(Private::ExceptionDestination::StdErr);
00044 }
```

TerminalInitializer() [1/3]

```
Term::TerminalInitializer::TerminalInitializer ( )  [noexcept]
```

Definition at line 20 of file terminal_initializer.cpp.

```
00022 {
00023   if(0 == m_counter)
00024   {
00025     static const Private::FileInitializer files_init;
00026     new(&Term::terminal) Terminal();
00027   }
00028   ++m_counter;
00029 }
00030 catch(...)
00031 {
00032   ExceptionHandler(Private::ExceptionDestination::StdErr);
00033 }
```

TerminalInitializer() [2/3]

```
Term::TerminalInitializer::TerminalInitializer (
            const TerminalInitializer &  )  [delete]
```

TerminalInitializer() [3/3]

```
Term::TerminalInitializer::TerminalInitializer (
            TerminalInitializer &&  )  [delete]
```

8.28.3 Member Function Documentation

operator=() [1/2]

```
TerminalInitializer & Term::TerminalInitializer::operator= (
            const TerminalInitializer &  )  [delete]
```

operator=() [2/2]

```
TerminalInitializer & Term::TerminalInitializer::operator= (
            TerminalInitializer &&  )  [delete]
```

8.28.4 Member Data Documentation

m_counter

```
std::size_t Term::TerminalInitializer::m_counter {0}  [static], [private]
```

Definition at line 18 of file terminal_initializer.hpp.
```
00018 {
```

The documentation for this class was generated from the following files:

- cpp-terminal/terminal_initializer.hpp
- cpp-terminal/terminal_initializer.cpp

8.29 Term::Terminfo Class Reference

```
#include <cpp-terminal/terminfo.hpp>
```

Public Types

- enum class ColorMode : std::uint8_t {
  Unset , NoColor , Bit3 , Bit4 ,
  Bit8 , Bit24 }
- enum class Bool : std::uint8_t { UTF8 = 0 , Legacy , ControlSequences }
- enum class String : std::uint8_t { TermEnv , TermName , TermVersion }
- enum class Integer : std::uint8_t
- using Booleans = std::array<bool, BoolNumber>
- using Strings = std::array<std::string, StringNumber>
- using Integers = std::array<std::uint32_t, IntegerNumber>

Public Member Functions

- Terminfo ()

Static Public Member Functions

- static bool get (const Term::Terminfo::Bool &key)
- static std::uint32_t get (const Term::Terminfo::Integer &key)
- static std::string get (const Term::Terminfo::String &key)
- static ColorMode getColorMode ()

Static Private Member Functions

- static void check ()
- static void checkTermEnv ()
- static void checkTerminalName ()
- static void checkTerminalVersion ()
- static void checkColorMode ()
- static void checkUTF8 ()
- static void checkLegacy ()
- static void checkControlSequences ()
- static void set (const Term::Terminfo::Bool &key, const bool &value)
- static void set (const Term::Terminfo::Integer &key, const std::uint32_t &value)
- static void set (const Term::Terminfo::String &key, const std::string &value)

Static Private Attributes

- static const constexpr std::size_t BoolNumber {3}
- static const constexpr std::size_t StringNumber {3}
- static const constexpr std::size_t IntegerNumber {0}
- static ColorMode m_colorMode {ColorMode::Unset}
- static Booleans m_booleans {}
- static Integers m_integers {}
- static Strings m_strings {}

8.29.1 Detailed Description

Definition at line 19 of file terminfo.hpp.

8.29.2 Member Typedef Documentation

Booleans

```
using Term::Terminfo::Booleans = std::array<bool, BoolNumber>
```

Definition at line 65 of file terminfo.hpp.

Integers

```
using Term::Terminfo::Integers = std::array<std::uint32_t, IntegerNumber>
```

Definition at line 67 of file terminfo.hpp.

Strings

```
using Term::Terminfo::Strings = std::array<std::string, StringNumber>
```

Definition at line 66 of file terminfo.hpp.

8.29.3 Member Enumeration Documentation

Bool

```
enum class Term::Terminfo::Bool :  std::uint8_t  [strong]
```

Enumerator

UTF8	terminal has UTF-8 activated.
Legacy	Terminal is in legacy mode (Windows only).
ControlSequences	Terminal support control sequences.

Definition at line 38 of file terminfo.hpp.

```
00039   {
00040     UTF8 = 0,
00041     Legacy,
00042     ControlSequences,
00043   };
```

ColorMode

```
enum class Term::Terminfo::ColorMode :  std::uint8_t  [strong]
```

Enumerator

Unset	
NoColor	
Bit3	
Bit4	
Bit8	
Bit24	

Definition at line 24 of file terminfo.hpp.

```
00025   {
00026     Unset,
00027     // no color was used
00028     NoColor,
00029     // a 3bit color was used
00030     Bit3,
00031     // a 4bit color was used
00032     Bit4,
00033     // a 8bit color was used
00034     Bit8,
00035     // a 24bit (RGB) color was used
00036     Bit24
00037   };
```

Integer

```
enum class Term::Terminfo::Integer :  std::uint8_t  [strong]
```

Definition at line 50 of file terminfo.hpp.

```
00051   {
00052
00053   };
```

String

```
enum class Term::Terminfo::String :  std::uint8_t  [strong]
```

Enumerator

TermEnv	TERM environment variable value.
TermName	Name of the terminal program if available.
TermVersion	Terminal version.

Definition at line 44 of file terminfo.hpp.

```
00045   {
00046     TermEnv,
```

```
00047     TermName,
00048     TermVersion,
00049   };
```

8.29.4 Constructor & Destructor Documentation

Terminfo()

```
Term::Terminfo::Terminfo ( )
```

Definition at line 139 of file terminfo.cpp.

```
00139 { check(); }
```

8.29.5 Member Function Documentation

check()

```
void Term::Terminfo::check ( )  [static], [private]
```

Definition at line 117 of file terminfo.cpp.

```
00118 {
00119   static bool checked{false};
00120   if(!checked)
00121   {
00122     checkTermEnv();
00123     checkTerminalName();
00124     checkTerminalVersion();
00125     checkControlSequences();
00126     checkLegacy();
00127     checkColorMode();
00128     checkUTF8();
00129     checked = true;
00130   }
00131 }
```

checkColorMode()

```
void Term::Terminfo::checkColorMode ( )  [static], [private]
```

Definition at line 141 of file terminfo.cpp.

```
00142 {
00143   std::string name{m_strings[static_cast<std::size_t>(Terminfo::String::TermName)]};
00144   if(name == "Apple_Terminal") { m_colorMode = Term::Terminfo::ColorMode::Bit8; }
00145   else if(name == "JetBrains-JediTerm") { m_colorMode = Term::Terminfo::ColorMode::Bit24; }
00146   else if(name == "vscode") { m_colorMode = Term::Terminfo::ColorMode::Bit24; }
00147   else if(name == "linux") { m_colorMode = Term::Terminfo::ColorMode::Bit4; }
00148   else if(name == "ansicon") { m_colorMode = Term::Terminfo::ColorMode::Bit4; }
00149   else if(m_strings[static_cast<std::size_t>(Terminfo::String::TermEnv)] == "linux") { m_colorMode =
      Term::Terminfo::ColorMode::Bit4; }
00150 #if defined(_WIN32)
00151   else if(WindowsVersionGreater(10, 0, 10586) &&
      !m_booleans[static_cast<std::size_t>(Terminfo::Bool::Legacy)]) { m_colorMode =
      Term::Terminfo::ColorMode::Bit24; }
00152   else if(m_booleans[static_cast<std::size_t>(Terminfo::Bool::Legacy)]) { m_colorMode =
      Term::Terminfo::ColorMode::Bit4; }
00153 #endif
00154   else { m_colorMode = Term::Terminfo::ColorMode::Bit24; }
00155   std::string colorterm = Private::getenv("COLORTERM").second;
00156   if((colorterm == "truecolor" || colorterm == "24bit") && m_colorMode != ColorMode::Unset) {
      m_colorMode = Term::Terminfo::ColorMode::Bit24; }
00157 }
```

checkControlSequences()

```
void Term::Terminfo::checkControlSequences ( )  [static], [private]
```

Definition at line 159 of file terminfo.cpp.

```
00160 {
00161 #ifdef _WIN32
00162   if(WindowsVersionGreater(10, 0, 10586)) { set(Term::Terminfo::Bool::ControlSequences, true); }
00163   else { set(Term::Terminfo::Bool::ControlSequences, false); }
00164 #else
00165   set(Term::Terminfo::Bool::ControlSequences, true);
00166 #endif
00167 }
```

checkLegacy()

```
void Term::Terminfo::checkLegacy ( )  [static], [private]
```

Definition at line 81 of file terminfo.cpp.

```
00082 {
00083 #if defined(_WIN32)
00084   #ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING
00085     #define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
00086   #endif
00087   if(!m_booleans[static_cast<std::size_t>(Terminfo::Bool::ControlSequences)]) {
      set(Terminfo::Bool::Legacy, true); }
00088   else
00089   {
00090     DWORD dwOriginalOutMode{0};
00091     GetConsoleMode(Private::out.handle(), &dwOriginalOutMode);
00092     if(!SetConsoleMode(Private::out.handle(), dwOriginalOutMode | ENABLE_VIRTUAL_TERMINAL_PROCESSING))
      { set(Terminfo::Bool::Legacy, true); }
00093     else
00094     {
00095       SetConsoleMode(Private::out.handle(), dwOriginalOutMode);
00096       set(Terminfo::Bool::Legacy, false);
00097     }
00098   }
00099 #else
00100   set(Terminfo::Bool::Legacy, false);
00101 #endif
00102 }
```

checkTermEnv()

```
void Term::Terminfo::checkTermEnv ( )  [static], [private]
```

Definition at line 104 of file terminfo.cpp.

```
00104 { set(Terminfo::String::TermEnv, Private::getenv("TERM").second); }
```

checkTerminalName()

```
void Term::Terminfo::checkTerminalName ( )  [static], [private]
```

Definition at line 106 of file terminfo.cpp.

```
00107 {
00108   std::string name;
00109   name = Private::getenv("TERM_PROGRAM").second;
00110   if(name.empty()) { name = Private::getenv("TERMINAL_EMULATOR").second; }
00111   if(Private::getenv("ANSICON").first) { name = "ansicon"; }
00112   set(Term::Terminfo::String::TermName, name);
00113 }
```

checkTerminalVersion()

```
void Term::Terminfo::checkTerminalVersion ( )  [static], [private]
```

Definition at line 115 of file terminfo.cpp.

```
00115 { set(Terminfo::String::TermVersion, Private::getenv("TERM_PROGRAM_VERSION").second); }
```

checkUTF8()

```
void Term::Terminfo::checkUTF8 ( )  [static], [private]
```

Definition at line 169 of file terminfo.cpp.

```
00170 {
00171 #if defined(_WIN32)
00172   (GetConsoleOutputCP() == CP_UTF8 && GetConsoleCP() == CP_UTF8) ? set(Terminfo::Bool::UTF8, true) :
      set(Terminfo::Bool::UTF8, false);
00173 #else
00174   Term::Cursor cursor_before{Term::cursor_position()};
00175   Term::Private::out.write("\xe2\x82\xac");  // € 3bits in utf8 one character
00176   std::string  read{Term::Private::in.read()};
00177   Term::Cursor cursor_after{Term::cursor_position()};
00178   std::size_t  moved{cursor_after.column() - cursor_before.column()};
00179   if(moved == 1) { set(Terminfo::Bool::UTF8, true); }
00180   else { set(Terminfo::Bool::UTF8, false); }
00181   for(std::size_t i = 0; i != moved; ++i) { Term::Private::out.write("\b \b"); }
00182 #endif
00183 }
```

get() [1/3]

```
bool Term::Terminfo::get (
            const Term::Terminfo::Bool & key )  [static]
```

Definition at line 27 of file terminfo.cpp.

```
00028 {
00029   check();
00030   return m_booleans[static_cast<std::size_t>(key)];
00031 }
```

get() [2/3]

```
std::uint32_t Term::Terminfo::get (
            const Term::Terminfo::Integer & key )  [static]
```

Definition at line 33 of file terminfo.cpp.

```
00034 {
00035   check();
00036   return m_integers[static_cast<std::size_t>(key)];
00037 }
```

get() [3/3]

```
std::string Term::Terminfo::get (
            const Term::Terminfo::String & key )  [static]
```

Definition at line 39 of file terminfo.cpp.

```
00040 {
00041   check();
00042   return m_strings[static_cast<std::size_t>(key)];
00043 }
```

getColorMode()

```
Term::Terminfo::ColorMode Term::Terminfo::getColorMode ( )  [static]
```

Definition at line 133 of file terminfo.cpp.

```
00134 {
00135   checkColorMode();
00136   return m_colorMode;
00137 }
```

set() [1/3]

```
void Term::Terminfo::set (
            const Term::Terminfo::Bool & key,
            const bool & value )  [static], [private]
```

Definition at line 45 of file terminfo.cpp.

```
00045 { m_booleans[static_cast<std::size_t>(key)] = value; }
```

set() [2/3]

```
void Term::Terminfo::set (
            const Term::Terminfo::Integer & key,
            const std::uint32_t & value )  [static], [private]
```

Definition at line 46 of file terminfo.cpp.

```
00046 { m_integers[static_cast<std::size_t>(key)] = value; }
```

set() [3/3]

```
void Term::Terminfo::set (
            const Term::Terminfo::String & key,
            const std::string & value )  [static], [private]
```

Definition at line 47 of file terminfo.cpp.

```
00047 { m_strings[static_cast<std::size_t>(key)] = value; }
```

8.29.6 Member Data Documentation

BoolNumber

```
const constexpr std::size_t Term::Terminfo::BoolNumber {3}  [static], [constexpr], [private]
```

Definition at line 60 of file terminfo.hpp.

```
00060 {3};
```

IntegerNumber

```
const constexpr std::size_t Term::Terminfo::IntegerNumber {0}  [static], [constexpr], [private]
```

Definition at line 62 of file terminfo.hpp.

```
00062 {0};
```

m_booleans

```
Term::Terminfo::Booleans Term::Terminfo::m_booleans {}  [static], [private]
```

Definition at line 23 of file terminfo.hpp.

m_colorMode

```
Term::Terminfo::ColorMode Term::Terminfo::m_colorMode {ColorMode::Unset}  [static], [private]
```

Definition at line 22 of file terminfo.hpp.

m_integers

```
Term::Terminfo::Integers Term::Terminfo::m_integers {}  [static], [private]
```

Definition at line 24 of file terminfo.hpp.

```
00024 : std::uint8_t
```

m_strings

```
Term::Terminfo::Strings Term::Terminfo::m_strings {}  [static], [private]
```

Definition at line 25 of file terminfo.hpp.

```
00025 {
```

StringNumber

```
const constexpr std::size_t Term::Terminfo::StringNumber {3}  [static], [constexpr], [private]
```

Definition at line 61 of file terminfo.hpp.

```
00061 {3};
```

The documentation for this class was generated from the following files:

- cpp-terminal/terminfo.hpp
- cpp-terminal/private/terminfo.cpp

8.30 Term::TIstream Class Reference

```
#include <cpp-terminal/stream.hpp>
```

Public Member Functions

- TIstream (const Term::Buffer::Type &type=Term::Buffer::Type::LineBuffered, const std::streamsize &size=BUFSIZ)
- TIstream (const TIstream &)=delete
- TIstream (TIstream &&other)=delete
- TIstream & operator= (TIstream &&)=delete
- TIstream & operator= (const TIstream &)=delete
- ~TIstream ()
- std::streambuf * rdbuf () const
- template<typename T >
  TIstream & operator>> (T &t)

Private Attributes

- Term::Buffer m_buffer
- std::istream m_stream

8.30.1 Detailed Description

Definition at line 20 of file stream.hpp.

8.30.2 Constructor & Destructor Documentation

TIstream() [1/3]

```
Term::TIstream::TIstream (
            const Term::Buffer::Type & type = Term::Buffer::Type::LineBuffered,
            const std::streamsize & size = BUFSIZ )  [explicit]
```

Definition at line 12 of file stream.cpp.

```
00012 : m_buffer(type, size), m_stream(&m_buffer) {}
```

TIstream() [2/3]

```
Term::TIstream::TIstream (
            const TIstream &  )  [delete]
```

TIstream() [3/3]

```
Term::TIstream::TIstream (
            TIstream && other )  [delete]
```

~TIstream()

```
Term::TIstream::~TIstream ( )
```

Definition at line 14 of file stream.cpp.

```
00014 { m_stream.clear(); }
```

8.30.3 Member Function Documentation

operator=() [1/2]

```
TIstream & Term::TIstream::operator= (
            const TIstream &  )  [delete]
```

operator=() [2/2]

```
TIstream & Term::TIstream::operator= (
            TIstream &&  )  [delete]
```

operator>>()

```
template<typename T >
TIstream & Term::TIstream::operator>> (
            T & t )  [inline]
```

Definition at line 30 of file stream.hpp.

```
00031   {
00032     m_stream » t;
00033     return *this;
00034   }
```

rdbuf()

```
std::streambuf * Term::TIstream::rdbuf ( ) const
```

Definition at line 16 of file stream.cpp.

```
00016 { return const_cast<Term::Buffer*>(&m_buffer); }
```

8.30.4 Member Data Documentation

m_buffer

```
Term::Buffer Term::TIstream::m_buffer  [private]
```

Definition at line 37 of file stream.hpp.

m_stream

```
std::istream Term::TIstream::m_stream  [private]
```

Definition at line 38 of file stream.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/stream.hpp
- cpp-terminal/stream.cpp

8.31 Term::TOstream Class Reference

`#include <cpp-terminal/stream.hpp>`

Public Member Functions

- TOstream (const Term::Buffer::Type &type=Term::Buffer::Type::LineBuffered, const std::streamsize &size=BUFSIZ)
- ~TOstream ()
- TOstream (const TOstream &)=delete
- TOstream (TOstream &&)=delete
- TOstream & operator= (TOstream &&)=delete
- TOstream & operator= (const TOstream &)=delete
- template<typename T >
  TOstream & operator<< (const T &t)
- TOstream & operator<< (std::ostream &(*t)(std::ostream &))

Private Attributes

- Term::Buffer m_buffer
- std::ostream m_stream

8.31.1 Detailed Description

Definition at line 41 of file stream.hpp.

8.31.2 Constructor & Destructor Documentation

TOstream() [1/3]

```
Term::TOstream::TOstream (
            const Term::Buffer::Type & type = Term::Buffer::Type::LineBuffered,
            const std::streamsize & size = BUFSIZ )  [explicit]
```

Definition at line 18 of file stream.cpp.

```
00018 : m_buffer(type, size), m_stream(&m_buffer) {}
```

~TOstream()

```
Term::TOstream::~TOstream ( )
```

Definition at line 20 of file stream.cpp.

```
00020 { m_stream.flush(); }
```

TOstream() [2/3]

```
Term::TOstream::TOstream (
            const TOstream &  )  [delete]
```

TOstream() [3/3]

```
Term::TOstream::TOstream (
            TOstream &&  )  [delete]
```

8.31.3 Member Function Documentation

operator<<() [1/2]

```
template<typename T >
TOstream & Term::TOstream::operator<< (
            const T & t )  [inline]
```

Definition at line 50 of file stream.hpp.

```
00051   {
00052     m_stream << t;
00053     return *this;
00054   }
```

operator<<() [2/2]

```
TOstream & Term::TOstream::operator<< (
            std::ostream &(*)(std::ostream &) t )  [inline]
```

Definition at line 55 of file stream.hpp.

```
00056   {
00057     m_stream << t;
00058     return *this;
00059   }
```

operator=() [1/2]

```
TOstream & Term::TOstream::operator= (
            const TOstream &  )  [delete]
```

operator=() [2/2]

```
TOstream & Term::TOstream::operator= (
            TOstream &&  )  [delete]
```

8.31.4 Member Data Documentation

m_buffer

```
Term::Buffer Term::TOstream::m_buffer  [private]
```

Definition at line 62 of file stream.hpp.

m_stream

```
std::ostream Term::TOstream::m_stream  [private]
```

Definition at line 63 of file stream.hpp.

The documentation for this class was generated from the following files:

- cpp-terminal/stream.hpp
- cpp-terminal/stream.cpp

8.32 Term::Window Class Reference

Represents a rectangular window, as a 2D array of characters and their attributes.

```
#include <cpp-terminal/window.hpp>
```

Public Member Functions

- Window (const std::size_t &columns, const std::size_t &rows)
- std::size_t get_w () const
- std::size_t get_h () const
- void set_char (const std::size_t &column, const std::size_t &row, const char32_t &character)
- void set_fg_reset (const std::size_t &column, const std::size_t &row)
- void set_bg_reset (const std::size_t &column, const std::size_t &row)
- void set_fg (const std::size_t &column, const std::size_t &row, const Color &color)
- void set_bg (const std::size_t &column, const std::size_t &row, const Color &color)
- void set_style (const std::size_t &column, const std::size_t &row, const Style &style)
- void set_cursor_pos (const std::size_t &column, const std::size_t &row)
- void set_h (const std::size_t &)
- void print_str (const std::size_t &column, const std::size_t &, const std::string &, const std::size_t &=0, bool=false)
- void fill_fg (const std::size_t &column, const std::size_t &, const std::size_t &, const std::size_t &, const Color &)
- void fill_bg (const std::size_t &column, const std::size_t &, const std::size_t &, const std::size_t &, const Color &)
- void fill_style (const std::size_t &column, const std::size_t &, const std::size_t &, const std::size_t &, const Style &)
- void print_border ()
- void print_rect (const std::size_t &column, const std::size_t &, const std::size_t &, const std::size_t &)
- void clear ()
- bool insideWindow (const std::size_t &column, const std::size_t &row) const
- std::string render (const std::size_t &, const std::size_t &, bool)

Private Member Functions

- std::size_t index (const std::size_t &column, const std::size_t &row) const
- char32_t get_char (const std::size_t &column, const std::size_t &row)
- bool get_fg_reset (const std::size_t &column, const std::size_t &row)
- bool get_bg_reset (const std::size_t &column, const std::size_t &row)
- Term::Color get_fg (const std::size_t &column, const std::size_t &row)
- Term::Color get_bg (const std::size_t &column, const std::size_t &row)
- Term::Style get_style (const std::size_t &column, const std::size_t &row)

Private Attributes

- Term::Screen m_window {0, 0}
- Term::Cursor m_cursor {1, 1}
- std::vector< char32_t > m_chars
- std::vector< Term::Color > m_fg
- std::vector< Term::Color > m_bg
- std::vector< bool > m_fg_reset
- std::vector< bool > m_bg_reset
- std::vector< Style > m_style

8.32.1 Detailed Description

Represents a rectangular window, as a 2D array of characters and their attributes.

Represents a rectangular window, as a 2D array of characters and their attributes. The render method can convert this internal representation to a string that when printed will show the Window on the screen. The natural way to represent a character in a terminal would be a "unicode grapheme cluster", but due to a lack of a good library for C++ that could handle those, we simply use a Unicode code point as a character.

Definition at line 32 of file window.hpp.

8.32.2 Constructor & Destructor Documentation

Window()

```
Term::Window::Window (
            const std::size_t & columns,
            const std::size_t & rows )
```

Definition at line 26 of file window.cpp.

```
00026 : m_window({rows, columns}) { clear(); }
```

8.32.3 Member Function Documentation

clear()

```
void Term::Window::clear ( )
```

Definition at line 187 of file window.cpp.

```
00188 {
00189   const std::size_t area{m_window.rows() * m_window.columns()};
00190   m_style.assign(area, Style::Reset);
00191   m_bg_reset.assign(area, true);
00192   m_bg.assign(area, Term::Color::Name::Default);
00193   m_fg_reset.assign(area, true);
00194   m_fg.assign(area, Term::Color::Name::Default);
00195   m_chars.assign(area, ' ');
00196 }
```

fill_bg()

```
void Term::Window::fill_bg (
              const std::size_t & column,
              const std::size_t & y1,
              const std::size_t & x2,
              const std::size_t & y2,
              const Color & rgb )
```

Definition at line 130 of file window.cpp.

```
00131 {
00132   for(std::size_t j = y1; j <= y2; ++j)
00133   {
00134     for(std::size_t i = x1; i <= x2; ++i) { set_bg(i, j, rgb); }
00135   }
00136 }
```

fill_fg()

```
void Term::Window::fill_fg (
              const std::size_t & column,
              const std::size_t & y1,
              const std::size_t & x2,
              const std::size_t & y2,
              const Color & rgb )
```

Definition at line 122 of file window.cpp.

```
00123 {
00124   for(std::size_t j = y1; j <= y2; ++j)
00125   {
00126     for(std::size_t i = x1; i <= x2; ++i) { set_fg(i, j, rgb); }
00127   }
00128 }
```

fill_style()

```
void Term::Window::fill_style (
              const std::size_t & column,
              const std::size_t & y1,
              const std::size_t & x2,
              const std::size_t & y2,
              const Style & color )
```

Definition at line 138 of file window.cpp.

```
00139 {
00140   for(std::size_t j = y1; j <= y2; ++j)
00141   {
00142     for(std::size_t i = x1; i <= x2; ++i) { set_style(i, j, color); }
00143   }
00144 }
```

get_bg()

```
Term::Color Term::Window::get_bg (
              const std::size_t & column,
              const std::size_t & row )  [private]
```

Definition at line 36 of file window.cpp.

```
00036 { return m_bg[index(column, row)]; }
```

get_bg_reset()

```
bool Term::Window::get_bg_reset (
            const std::size_t & column,
            const std::size_t & row )  [private]
```

Definition at line 32 of file window.cpp.

```
00032 { return m_bg_reset[index(column, row)]; }
```

get_char()

```
char32_t Term::Window::get_char (
            const std::size_t & column,
            const std::size_t & row )  [private]
```

Definition at line 28 of file window.cpp.

```
00028 { return m_chars[index(column, row)]; }
```

get_fg()

```
Term::Color Term::Window::get_fg (
            const std::size_t & column,
            const std::size_t & row )  [private]
```

Definition at line 34 of file window.cpp.

```
00034 { return m_fg[index(column, row)]; }
```

get_fg_reset()

```
bool Term::Window::get_fg_reset (
            const std::size_t & column,
            const std::size_t & row )  [private]
```

Definition at line 30 of file window.cpp.

```
00030 { return m_fg_reset[index(column, row)]; }
```

get_h()

```
std::size_t Term::Window::get_h ( ) const
```

Definition at line 42 of file window.cpp.

```
00042 { return m_window.rows(); }
```

get_style()

```
Term::Style Term::Window::get_style (
            const std::size_t & column,
            const std::size_t & row )  [private]
```

Definition at line 38 of file window.cpp.

```
00038 { return m_style[index(column, row)]; }
```

get_w()

```
std::size_t Term::Window::get_w ( ) const
```

Definition at line 40 of file window.cpp.

```
00040 { return m_window.columns(); }
```

index()

```
std::size_t Term::Window::index (
            const std::size_t & column,
            const std::size_t & row ) const  [private]
```

Definition at line 298 of file window.cpp.

```
00299 {
00300   if(!insideWindow(column, row)) { throw Term::Exception("Cursor out of range"); }
00301   return ((row - 1) * m_window.columns()) + (column - 1);
00302 }
```

insideWindow()

```
bool Term::Window::insideWindow (
            const std::size_t & column,
            const std::size_t & row ) const
```

Definition at line 304 of file window.cpp.

```
00304 { return (column >= 1) && (row >= 1) && (column <= m_window.columns()) && (row <= m_window.rows()); }
```

print_border()

```
void Term::Window::print_border ( )
```

Definition at line 146 of file window.cpp.

```
00146 { print_rect(1, 1, m_window.columns(), m_window.rows()); }
```

print_rect()

```
void Term::Window::print_rect (
            const std::size_t & column,
            const std::size_t & y1,
            const std::size_t & x2,
            const std::size_t & y2 )
```

Definition at line 148 of file window.cpp.

```
00149 {
00150   std::u32string border = Private::utf8_to_utf32("│─┌┐└┘");
00151   if(Term::Terminfo::get(Term::Terminfo::Bool::UTF8))
00152   {
00153     for(std::size_t j = y1 + 1; j <= (y2 - 1); ++j)
00154     {
00155       set_char(x1, j, border[0]);
00156       set_char(x2, j, border[0]);
00157     }
00158     for(std::size_t i = x1 + 1; i <= (x2 - 1); ++i)
00159     {
00160       set_char(i, y1, border[1]);
00161       set_char(i, y2, border[1]);
00162     }
```

```
00163     set_char(x1, y1, border[2]);
00164     set_char(x2, y1, border[3]);
00165     set_char(x1, y2, border[4]);
00166     set_char(x2, y2, border[5]);
00167   }
00168   else
00169   {
00170     for(std::size_t j = y1 + 1; j <= (y2 - 1); ++j)
00171     {
00172       set_char(x1, j, '|');
00173       set_char(x2, j, '|');
00174     }
00175     for(std::size_t i = x1 + 1; i <= (x2 - 1); ++i)
00176     {
00177       set_char(i, y1, '-');
00178       set_char(i, y2, '-');
00179     }
00180     set_char(x1, y1, '+');
00181     set_char(x2, y1, '+');
00182     set_char(x1, y2, '+');
00183     set_char(x2, y2, '+');
00184   }
00185 }
```

print_str()

```
void Term::Window::print_str (
              const std::size_t & column,
              const std::size_t & y,
              const std::string & s,
              const std::size_t & indent = 0,
              bool move_cursor = false )
```

Definition at line 95 of file window.cpp.

```
00096 {
00097   std::u32string s2   = Private::utf8_to_utf32(s);
00098   std::size_t    xpos = x;
00099   std::size_t    ypos = y;
00100   for(char32_t i: s2)
00101   {
00102     if(i == U'\n')
00103     {
00104       xpos = x + indent;
00105       ypos++;
00106       if(insideWindow(xpos, ypos))
00107       {
00108         for(std::size_t j = 0; j < indent; ++j) { set_char(x + j, ypos, '.'); }
00109       }
00110       else { return; }
00111     }
00112     else
00113     {
00114       if(insideWindow(xpos, ypos)) { set_char(xpos, y, i); }
00115       else { return; }
00116       ++xpos;
00117     }
00118   }
00119   if(move_cursor) { m_cursor = {ypos, xpos}; }
00120 }
```

render()

```
std::string Term::Window::render (
              const std::size_t & x0,
              const std::size_t & y0,
              bool term )
```

Definition at line 198 of file window.cpp.

```
00199 {
00200   std::string out;
00201   if(term) { out.append(cursor_off()); }
00202   Color current_fg       = Term::Color::Name::Default;
```

```
00203   Color current_bg       = Term::Color::Name::Default;
00204   bool  current_fg_reset = true;
00205   bool  current_bg_reset = true;
00206   Style current_style    = Style::Reset;
00207   for(std::size_t j = 1; j <= m_window.rows(); ++j)
00208   {
00209     if(term) { out.append(cursor_move(y0 + j - 1, x0)); }
00210     for(std::size_t i = 1; i <= m_window.columns(); ++i)
00211     {
00212       bool update_fg       = false;
00213       bool update_bg       = false;
00214       bool update_fg_reset = false;
00215       bool update_bg_reset = false;
00216       bool update_style    = false;
00217       if(current_fg_reset != get_fg_reset(i, j))
00218       {
00219         current_fg_reset = get_fg_reset(i, j);
00220         if(current_fg_reset)
00221         {
00222           update_fg_reset = true;
00223           current_fg      = {255, 255, 255};
00224         }
00225       }
00226
00227       if(current_bg_reset != get_bg_reset(i, j))
00228       {
00229         current_bg_reset = get_bg_reset(i, j);
00230         if(current_bg_reset)
00231         {
00232           update_bg_reset = true;
00233           current_bg      = {255, 255, 255};
00234         }
00235       }
00236
00237       if(!current_fg_reset)
00238       {
00239         if(!(current_fg == get_fg(i, j)))
00240         {
00241           current_fg = get_fg(i, j);
00242           update_fg  = true;
00243         }
00244       }
00245
00246       if(!current_fg_reset)
00247       {
00248         if(!(current_bg == get_bg(i, j)))
00249         {
00250           current_bg = get_bg(i, j);
00251           update_bg  = true;
00252         }
00253       }
00254       if(current_style != get_style(i, j))
00255       {
00256         current_style = get_style(i, j);
00257         update_style  = true;
00258         if(current_style == Style::Reset)
00259         {
00260           // style::reset: reset fg and bg colors too, we have to
00261           // set them again if they are non-default, but if fg or
00262           // bg colors are reset, we do not update them, as
00263           // style::reset already did that.
00264           update_fg = !current_fg_reset;
00265           update_bg = !current_bg_reset;
00266         }
00267       }
00268       // Set style first, as style::reset will reset colors too
00269       if(update_style) { out.append(style(get_style(i, j))); }
00270       if(update_fg_reset) { out.append(color_fg(Term::Color::Name::Default)); }
00271       else if(update_fg)
00272       {
00273         const Term::Color color_tmp = get_fg(i, j);
00274         out.append(color_fg(color_tmp));
00275       }
00276
00277       if(update_bg_reset) { out.append(color_bg(Term::Color::Name::Default)); }
00278       else if(update_bg)
00279       {
00280         const Term::Color color_tmp = get_bg(i, j);
00281         out.append(color_bg(color_tmp));
00282       }
00283       out.append(Private::utf32_to_utf8(get_char(i, j)));
00284     }
00285     if(j < m_window.rows()) { out.append("\n"); }
00286   }
00287   if(!current_fg_reset) { out.append(color_fg(Term::Color::Name::Default)); }
00288   if(!current_bg_reset) { out.append(color_bg(Term::Color::Name::Default)); }
00289   if(current_style != Style::Reset) { out.append(style(Style::Reset)); }
```

```
00290   if(term)
00291   {
00292     out.append(cursor_move(y0 + (m_cursor.row() - 1), x0 + (m_cursor.column() - 1)));
00293     out.append(cursor_on());
00294   }
00295   return out;
00296 }
```

set_bg()

```
void Term::Window::set_bg (
              const std::size_t & column,
              const std::size_t & row,
              const Color & color )
```

Definition at line 68 of file window.cpp.

```
00069 {
00070   m_bg_reset[index(column, row)] = false;
00071   m_bg[index(column, row)]       = color;
00072 }
```

set_bg_reset()

```
void Term::Window::set_bg_reset (
              const std::size_t & column,
              const std::size_t & row )
```

Definition at line 56 of file window.cpp.

```
00057 {
00058   m_bg_reset[index(column, row)] = true;
00059   m_bg[index(column, row)]       = Term::Color::Name::Default;
00060 }
```

set_char()

```
void Term::Window::set_char (
              const std::size_t & column,
              const std::size_t & row,
              const char32_t & character )
```

Definition at line 44 of file window.cpp.

```
00045 {
00046   if(insideWindow(column, row)) { m_chars[index(column, row)] = character; }
00047   else { throw Term::Exception("set_char(): (x,y) out of bounds"); }
00048 }
```

set_cursor_pos()

```
void Term::Window::set_cursor_pos (
              const std::size_t & column,
              const std::size_t & row )
```

Definition at line 76 of file window.cpp.

```
00076 { m_cursor = {row, column}; }
```

set_fg()

```
void Term::Window::set_fg (
              const std::size_t & column,
              const std::size_t & row,
              const Color & color )
```

Definition at line 62 of file window.cpp.

```
00063 {
00064   m_fg_reset[index(column, row)] = false;
00065   m_fg[index(column, row)]       = color;
00066 }
```

set_fg_reset()

```
void Term::Window::set_fg_reset (
              const std::size_t & column,
              const std::size_t & row )
```

Definition at line 50 of file window.cpp.

```
00051 {
00052   m_fg_reset[index(column, row)] = true;
00053   m_fg[index(column, row)]       = Term::Color::Name::Default;
00054 }
```

set_h()

```
void Term::Window::set_h (
              const std::size_t & new_h )
```

Definition at line 78 of file window.cpp.

```
00079 {
00080   if(new_h == m_window.rows()) { return; }
00081   if(new_h > m_window.rows())
00082   {
00083     const std::size_t dc = (new_h - m_window.rows()) * m_window.columns();
00084     m_chars.insert(m_chars.end(), dc, ' ');
00085     m_fg_reset.insert(m_fg_reset.end(), dc, true);
00086     m_bg_reset.insert(m_bg_reset.end(), dc, true);
00087     m_fg.insert(m_fg.end(), dc, {0, 0, 0});
00088     m_bg.insert(m_bg.end(), dc, {0, 0, 0});
00089     m_style.insert(m_style.end(), dc, Style::Reset);
00090     m_window = {m_window.columns(), new_h};
00091   }
00092   else { throw Term::Exception("Shrinking height not supported."); }
00093 }
```

set_style()

```
void Term::Window::set_style (
              const std::size_t & column,
              const std::size_t & row,
              const Style & style )
```

Definition at line 74 of file window.cpp.

```
00074 { m_style[index(column, row)] = style; }
```

8.32.4 Member Data Documentation

m_bg

```
std::vector<Term::Color> Term::Window::m_bg  [private]
```

Definition at line 85 of file window.hpp.

m_bg_reset

```
std::vector<bool> Term::Window::m_bg_reset  [private]
```

Definition at line 87 of file window.hpp.

m_chars

```
std::vector<char32_t> Term::Window::m_chars  [private]
```

Definition at line 83 of file window.hpp.

m_cursor

```
Term::Cursor Term::Window::m_cursor {1, 1}  [private]
```

Definition at line 82 of file window.hpp.
```
00082 {1, 1};
```

m_fg

```
std::vector<Term::Color> Term::Window::m_fg  [private]
```

Definition at line 84 of file window.hpp.

m_fg_reset

```
std::vector<bool> Term::Window::m_fg_reset  [private]
```

Definition at line 86 of file window.hpp.

m_style

```
std::vector<Style> Term::Window::m_style  [private]
```

Definition at line 88 of file window.hpp.

m_window

```
Term::Screen Term::Window::m_window {0, 0}  [private]
```

Definition at line 81 of file window.hpp.
```
00081 {0, 0};
```

The documentation for this class was generated from the following files:

- cpp-terminal/window.hpp
- cpp-terminal/window.cpp

8.33 Term::Private::WindowsError Class Reference

```
#include <cpp-terminal/private/exception.hpp>
```

Public Member Functions

- WindowsError (const WindowsError &)=default
- WindowsError (WindowsError &&)=default
- WindowsError () noexcept=default
- virtual ~WindowsError () noexcept=default
- WindowsError & operator= (WindowsError &&) noexcept=default
- WindowsError & operator= (const WindowsError &) noexcept=default
- std::int64_t error () const noexcept
- bool check_value () const noexcept
- WindowsError & check_if (const bool &ret) noexcept
- void throw_exception (const std::string &str=std::string()) const

Private Attributes

- std::int64_t m_error {0}
- bool m_check_value {false}

8.33.1 Detailed Description

Definition at line 24 of file exception.hpp.

8.33.2 Constructor & Destructor Documentation

WindowsError() [1/3]

```
Term::Private::WindowsError::WindowsError (
            const WindowsError &  )  [default]
```

WindowsError() [2/3]

```
Term::Private::WindowsError::WindowsError (
            WindowsError &&  )  [default]
```

WindowsError() [3/3]

```
Term::Private::WindowsError::WindowsError ( )  [default], [noexcept]
```

~WindowsError()

```
virtual Term::Private::WindowsError::~WindowsError ( )  [virtual], [default], [noexcept]
```

8.33.3 Member Function Documentation

check_if()

```
Term::Private::WindowsError & Term::Private::WindowsError::check_if (
            const bool & ret )  [noexcept]
```

Definition at line 71 of file exception.cpp.

```
00072 {
00073   m_error       = static_cast<std::int64_t>(GetLastError());
00074   m_check_value = ret;
00075   return *this;
00076 }
```

check_value()

```
bool Term::Private::WindowsError::check_value ( ) const  [noexcept]
```

Definition at line 69 of file exception.cpp.

```
00069 { return m_check_value; }
```

error()

```
std::int64_t Term::Private::WindowsError::error ( ) const  [noexcept]
```

Definition at line 67 of file exception.cpp.

```
00067 { return m_error; }
```

operator=() [1/2]

```
WindowsError & Term::Private::WindowsError::operator= (
            const WindowsError &  )  [default], [noexcept]
```

operator=() [2/2]

```
WindowsError & Term::Private::WindowsError::operator= (
            WindowsError &&  )  [default], [noexcept]
```

throw_exception()

```
void Term::Private::WindowsError::throw_exception (
            const std::string & str = std::string() ) const
```

Definition at line 78 of file exception.cpp.

```
00079 {
00080   if(m_check_value) { throw Term::Private::WindowsException(m_error, str); }
00081 }
```

8.33.4 Member Data Documentation

m_check_value

```
bool Term::Private::WindowsError::m_check_value {false}  [private]
```

Definition at line 40 of file exception.hpp.

```
00040 {false};
```

m_error

```
std::int64_t Term::Private::WindowsError::m_error {0}  [private]
```

Definition at line 39 of file exception.hpp.

```
00039 {0};
```

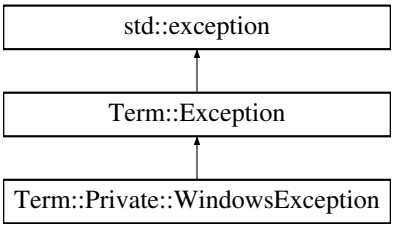
The documentation for this class was generated from the following files:

- cpp-terminal/private/exception.hpp
- cpp-terminal/private/exception.cpp

8.34 Term::Private::WindowsException Class Reference

```
#include <cpp-terminal/private/exception.hpp>
```

Inheritance diagram for Term::Private::WindowsException:



Public Member Functions

- WindowsException (const std::int64_t &error, const std::string &context=std::string())
- ~WindowsException () override=default

Public Member Functions inherited from Term::Exception

- Exception (const std::string &message) noexcept
- Exception (const std::int64_t &code, const std::string &message) noexcept
- Exception (const Exception &)=default
- Exception (Exception &&)=default
- Exception & operator= (Exception &&)=default
- Exception & operator= (const Exception &)=default
- const char * what () const noexcept override
- std::int64_t code () const noexcept
- std::string message () const noexcept
- std::string context () const noexcept
- ~Exception () noexcept override=default

Private Member Functions

- void build_what () const noexcept final

Additional Inherited Members

Protected Member Functions inherited from Term::Exception

- Exception (const std::int64_t &code) noexcept
- void setMessage (const std::string &message) noexcept
- void setContext (const std::string &context) noexcept
- void setWhat (const std::string &what) const noexcept

Static Protected Attributes inherited from Term::Exception

- static const constexpr std::size_t m_maxSize {256}

8.34.1 Detailed Description

Definition at line 43 of file exception.hpp.

8.34.2 Constructor & Destructor Documentation

WindowsException()

```
Term::Private::WindowsException::WindowsException (
            const std::int64_t & error,
            const std::string & context = std::string() )
```

Definition at line 83 of file exception.cpp.

```
00083                                                                                                 :
      Term::Exception(static_cast<std::int64_t>(error))
00084 {
00085   setContext(context);
00086   wchar_t*    ptr{nullptr};
00087   const DWORD cchMsg{FormatMessageW(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS |
      FORMAT_MESSAGE_ALLOCATE_BUFFER, nullptr, static_cast<uint32_t>(code()), MAKELANGID(LANG_ENGLISH,
      SUBLANG_ENGLISH_US), reinterpret_cast<wchar_t*>(&ptr), 0, nullptr)};
00088   if(cchMsg > 0)
00089   {
00090     auto deleter = [](void* p)
00091     {
00092       if(p != nullptr) { ::LocalFree(p); }
00093     };
00094     std::unique_ptr<wchar_t, decltype(deleter)> ptrBuffer(ptr, deleter);
00095     std::string                                 ret{Term::Private::to_narrow(ptrBuffer.get())};
00096     if(ret.size() >= 2 && ret[ret.size() - 1] == '\n' && ret[ret.size() - 2] == '\r')
      ret.erase(ret.size() - 2);
00097     setMessage(ret);
00098   }
00099   else { throw Term::Exception(::GetLastError(), "Error in FormatMessageW"); }
00100 }
```

~WindowsException()

```
Term::Private::WindowsException::~WindowsException ( )  [override], [default]
```

8.34.3 Member Function Documentation

build_what()

```
void Term::Private::WindowsException::build_what ( ) const  [final], [private], [virtual],
[noexcept]
```

Reimplemented from Term::Exception.

Definition at line 102 of file exception.cpp.

```
00103 {
00104   std::string what{std::string("windows error ") + std::to_string(code()) + std::string(": ") +
      message().c_str()};
00105   if(!context().empty()) what += " [" + context() + "]";
00106   setWhat(what);
00107 }
```

The documentation for this class was generated from the following files:

- cpp-terminal/private/exception.hpp
- cpp-terminal/private/exception.cpp

9 File Documentation

9.1 cpp-terminal/args.hpp File Reference

```
#include <string>
#include <vector>
```

Classes

- class Term::Arguments
- class Term::Argc

Namespaces

- namespace Term

9.2 args.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <string>
00013 #include <vector>
00014
00015 namespace Term
00016 {
00017
00018 class Arguments
00019 {
00020 public:
00021   Arguments();
00022   static std::size_t              argc();
00023   static std::vector<std::string> argv();
00024   std::string                     operator[](const std::size_t& arg) const;
00025
00026 private:
00027   static void                     parse();
00028   static std::vector<std::string> m_args;
00029   static bool                     m_parsed;
00030 };
00031
00032 class Argc
00033 {
00034 public:
00035   Argc();
00036   operator unsigned int();
00037   operator unsigned int() const;
00038 };
00039
00040 static const Arguments argv;
00041 static const Argc      argc;
00042
00043 }  // namespace Term
```

9.3 cpp-terminal/buffer.cpp File Reference

```
#include "cpp-terminal/buffer.hpp"
#include "cpp-terminal/options.hpp"
#include "cpp-terminal/private/file.hpp"
#include "cpp-terminal/terminal.hpp"
#include <cstddef>
```

9.4 buffer.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/buffer.hpp"
00011
00012 #include "cpp-terminal/options.hpp"
00013 #include "cpp-terminal/private/file.hpp"
00014 #include "cpp-terminal/terminal.hpp"
00015
00016 #include <cstddef>
00017
00018 static std::string replace(const Term::Buffer::int_type& c)
00019 {
00020 #if defined(_WIN32)
00021   std::string ret;
00022   if(static_cast<char>(c) == '\n') ret = "\r\n";
00023   else
00024     ret.push_back(static_cast<char>(c));
00025   return ret;
00026 #else
00027   std::string ret;
00028   ret.push_back(static_cast<char>(c));
00029   return ret;
00030 #endif
00031 }
00032
00033 static bool newline_sequence(const std::string& str)  //https://en.wikipedia.org/wiki/Newline
00034 {
00035   if(str.back() == '\n' || str.back() == '\r' || str.back() == '\036' || str.back() == '\036' ||
      str.back() == '\025') return true;
00036   else
00037     return false;
00038 }
00039
00040 int Term::Buffer::sync()
00041 {
00042   const int ret = Term::Private::out.write(m_buffer);
00043   m_buffer.clear();
00044   return ret;
00045 }
00046
00047 Term::Buffer::Buffer(const Term::Buffer::Type& type, const std::streamsize& size)
00048 {
00049   setType(type);
00050   switch(m_type)
00051   {
00052     case Type::Unbuffered: setbuf(nullptr, 0); break;
00053     case Type::LineBuffered:
00054     case Type::FullBuffered: setbuf(&m_buffer[0], size); break;
00055   }
00056 }
00057
00058 void Term::Buffer::setType(const Term::Buffer::Type& type) { m_type = type; }
00059
00060 std::streambuf* Term::Buffer::setbuf(char* s, std::streamsize n)
00061 {
00062   if(s != nullptr) m_buffer.reserve(static_cast<std::size_t>(n));
00063   return this;
00064 }
00065
```

```
00066 Term::Buffer::int_type Term::Buffer::underflow()
00067 {
00068   try
00069   {
00070     //TODO Maybe use input function ?
00071     m_buffer.clear();
00072     if(terminal.getOptions().has(Option::Raw))
00073     {
00074       do {
00075         std::string ret{Term::Private::in.read()};
00076         if(!ret.empty())
00077         {
00078           if(ret[0] == '\x7f' || ret[0] == '\b')
00079           {
00080             Term::Private::out.write("\b \b");  //Backspace is DEL, CTRL+Backspace is Backspace '\b'
00081             if(!m_buffer.empty()) m_buffer.erase(m_buffer.size() - 1);
00082           }
00083           else if(ret[0] == '\033')
00084           {
00085             continue;  // For now if it's escape sequence do nothing
00086           }
00087           else if(ret[0] <= 31 && ret[0] != '\t' && ret[0] != '\b' && ret[0] != 127 && ret[0] != ' '
      && ret[0] != '\n' && ret[0] != '\r') { continue; }
00088           else
00089           {
00090             Term::Private::out.write(ret);
00091             m_buffer += ret;
00092           }
00093         }
00094       } while(m_buffer.empty() || !newline_sequence(m_buffer));
00095       Term::Private::out.write('\n');
00096     }
00097     else
00098     {
00099       do {
00100         std::string ret{Term::Private::in.read()};
00101         m_buffer += ret;
00102       } while(m_buffer.empty());
00103     }
00104     setg(&m_buffer[0], &m_buffer[0], &m_buffer[0] + m_buffer.size());
00105     return traits_type::to_int_type(m_buffer.at(0));
00106   }
00107   catch(...)
00108   {
00109     return traits_type::eof();
00110   }
00111 }
00112
00113 Term::Buffer::int_type Term::Buffer::overflow(int c)
00114 {
00115   if(c != std::char_traits<Term::Buffer::char_type>::eof())
00116   {
00117     switch(m_type)
00118     {
00119       case Type::Unbuffered:
00120       {
00121         Term::Private::out.write(replace(c));
00122         break;
00123       }
00124       case Type::LineBuffered:
00125       {
00126         m_buffer += replace(c);
00127         if(static_cast<char>(c) == '\n')
00128         {
00129           Term::Private::out.write(m_buffer);
00130           m_buffer.clear();
00131         }
00132         break;
00133       }
00134       case Type::FullBuffered:
00135       {
00136         if(m_buffer.size() >= m_buffer.capacity())
00137         {
00138           Term::Private::out.write(m_buffer);
00139           m_buffer.clear();
00140         }
00141         m_buffer += replace(c);
00142         break;
00143       }
00144     }
00145   }
00146   return c;
00147 }
00148
00149 Term::Buffer::~Buffer()
00150 {
00151   //sync();
```

```
00152 }
```

9.5 cpp-terminal/buffer.hpp File Reference

```
#include <cstddef>
#include <cstdint>
#include <streambuf>
```

Classes

- class Term::Buffer

Namespaces

- namespace Term

9.6 buffer.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstddef>
00013 #include <cstdint>
00014 #include <streambuf>
00015
00016 namespace Term
00017 {
00018
00019 class Buffer final : public std::streambuf
00020 {
00021 public:
00022   enum class Type : std::uint8_t
00023   {
00024     Unbuffered,
00025     LineBuffered,
00026     FullBuffered
00027   };
00028   explicit Buffer(const Term::Buffer::Type& type = Term::Buffer::Type::LineBuffered, const
      std::streamsize& size = BUFSIZ);
00029   ~Buffer() override;
00030   Buffer(const Buffer&)            = delete;
00031   Buffer(Buffer&&)                 = delete;
00032   Buffer& operator=(Buffer&&)      = delete;
00033   Buffer& operator=(const Buffer&) = delete;
00034
00035 protected:
00036   int_type underflow() override;
00037   int_type overflow(int c = std::char_traits<Term::Buffer::char_type>::eof()) override;
00038   int      sync() override;
00039
00040 private:
00041   void               setType(const Term::Buffer::Type& type);
00042   std::streambuf*    setbuf(char* s, std::streamsize n) override;
00043   std::string        m_buffer;
00044   Term::Buffer::Type m_type{Term::Buffer::Type::LineBuffered};
00045 };
00046
00047 }  // namespace Term
```

9.7 cpp-terminal/color.cpp File Reference

```
#include "cpp-terminal/color.hpp"
#include "cpp-terminal/terminfo.hpp"
```

9.8 color.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/color.hpp"
00011
00012 #include "cpp-terminal/terminfo.hpp"
00013
00014 bool Term::Color::operator==(const Term::Color& color) const
00015 {
00016   if(color.getType() != getType()) { return false; }
00017   if(getType() == Term::Color::Type::Bit24) { return m_bit24 == color.to24bits(); }
00018   return m_bit8 == color.to8bits();
00019 }
00020
00021 bool Term::Color::operator!=(const Term::Color& color) const { return !(*this == color); }
00022
00023 Term::Color::Color() : m_bit8(0) {}
00024
00025 Term::Color::Color(const Term::Color::Name& name) : m_Type(Type::Bit4),
      m_bit8(static_cast<std::uint8_t>(name)) {}
00026
00027 Term::Color::Color(const std::uint8_t& color) : m_Type(Type::Bit8), m_bit8(color) {}
00028
00029 Term::Color::Color(const std::uint8_t& r, const std::uint8_t& b, const std::uint8_t& g) :
      m_Type(Type::Bit24)
00030 {
00031   // Hack for gcc4.7
00032   m_bit24[0] = r;
00033   m_bit24[1] = b;
00034   m_bit24[2] = g;
00035 }
00036
00037 Term::Color::Type Term::Color::getType() const { return m_Type; }
00038
00039 Term::Color::Name Term::Color::to3bits() const
00040 {
00041   if(getType() == Type::Bit3) { return static_cast<Term::Color::Name>(m_bit8); }
00042   const Term::Color::Name ret{to4bits()};
00043   if(ret >= Term::Color::Name::Gray) { return static_cast<Term::Color::Name>(static_cast<uint8_t>(ret)
      - static_cast<uint8_t>(Term::Color::Name::Gray)); }
00044   return ret;
00045 }
00046
00047 // Convert 24bits and 8bits to 4bits
00048 Term::Color::Name Term::Color::to4bits() const
00049 {
00050   //https://ajalt.github.io/colormath/converter/
00051   // clang-format off
00052   static const constexpr std::array<std::uint8_t,256> table ={{
00053     0,  1,  2,  3,  4,  5,  6,  7,  60, 61, 62, 63, 64, 65, 66, 67, 0,  4,  4,  4,  64, 64, 2,  6,  4,
      4,  64, 64, 2,  2,  6,  4,  64, 64, 2,  2,  2,  6,  64, 64, 62, 62, 62, 62, 66, 64, 62, 62, 62, 62,
      62, 66,
00054     1,  5,  4,  4,  64, 64, 3,  60, 4,  4,  64, 64, 2,  2,  6,  4,  64, 64, 2,  2,  2,  6,  64, 64,
      62, 62, 62, 62, 66, 64, 62, 62, 62, 62, 62, 66, 1,  1,  5,  4,  64, 64, 1,  1,  5,  4,  64, 64, 3,  3,
      60, 4,
00055     64, 64, 2,  2,  2,  6,  64, 64, 62, 62, 62, 62, 66, 64, 62, 62, 62, 62, 62, 66, 1,  1,  1,  5,
      64, 64, 1,  1,  1,  5,  64, 64, 1,  1,  1,  5,  64, 64, 3,  3,  3,  7,  64, 64, 62, 62, 62, 62, 66,
      64, 62, 62,
00056     62, 62, 62, 66, 61, 61, 61, 61, 65, 64, 61, 61, 61, 61, 65, 64, 61, 61, 61, 61, 65, 64, 61, 61,
      61, 61, 65, 64, 63, 63, 63, 63, 7,  64, 62, 62, 62, 62, 62, 66, 61, 61, 61, 61, 61, 65, 61, 61, 61,
      61, 61, 65,
00057     61, 61, 61, 61, 61, 65, 61, 61, 61, 61, 61, 65, 61, 61, 61, 61, 61, 65, 63, 63, 63, 63, 63, 67, 0,
      0,  0,  0,  0,  0,  60, 60, 60, 60, 60, 60, 7,  7,  7,  7,  7,  7,  67, 67, 67, 67, 67, 67}};
00058   // clang-format on
```

```
00059   if((getType() == Term::Color::Type::Bit24) || (getType() == Term::Color::Type::Bit8)) { return
      static_cast<Term::Color::Name>(table[to8bits()]); }
00060   return static_cast<Term::Color::Name>(m_bit8);
00061 }
00062
00063 // Convert 24bits to 8bits
00064 std::uint8_t Term::Color::to8bits() const
00065 {
00066   if(getType() == Term::Color::Type::Bit24)
00067   {
00068     // check gray scale in 24 steps
00069     if(m_bit24[0] == m_bit24[1] && m_bit24[0] == m_bit24[2]) { return static_cast<std::uint8_t>(232 +
      m_bit24[0] / 32 + m_bit24[1] / 32 + m_bit24[2] / 32); }
00070     // normal color space
00071     return static_cast<std::uint8_t>(16 + 36 * (m_bit24[0] / 51) + 6 * (m_bit24[1] / 51) + (m_bit24[2]
      / 51));
00072   }
00073   return m_bit8;
00074 }
00075
00076 // Nothing to do
00077 std::array<std::uint8_t, 3> Term::Color::to24bits() const { return m_bit24; }
00078
00079 std::string Term::color_bg(const Term::Color::Name& color) { return color_bg(Color(color)); }
00080
00081 std::string Term::color_bg(const std::uint8_t& color) { return color_bg(Color(color)); }
00082
00083 std::string Term::color_bg(const std::uint8_t& r, const std::uint8_t& g, const std::uint8_t& b) {
      return color_bg(Color(r, g, b)); }
00084
00085
      //https://unix.stackexchange.com/questions/212933/background-color-whitespace-when-end-of-the-terminal-reached
00086 //FIX maybe we need an other function without  [K if we want to modify background of part of the
      screen (Moving cursor and changing color )
00087 std::string Term::color_bg(const Color& color)
00088 {
00089   if(color.getType() == Term::Color::Type::Unset || color.getType() == Term::Color::Type::NoColor)
      return "";
00090   switch(Term::Terminfo::getColorMode())
00091   {
00092     case Term::Terminfo::ColorMode::Unset:
00093     case Term::Terminfo::ColorMode::NoColor: return {};
00094     case Term::Terminfo::ColorMode::Bit3: return "\u001b[" +
      std::to_string(static_cast<uint8_t>(color.to3bits()) + 40) + "m\u001b[K";
00095     case Term::Terminfo::ColorMode::Bit4: return "\u001b[" +
      std::to_string(static_cast<uint8_t>(color.to4bits()) + 40) + "m\u001b[K";
00096     case Term::Terminfo::ColorMode::Bit8:
00097       if(color.getType() == Term::Color::Type::Bit4 || color.getType() == Term::Color::Type::Bit3)
      return "\u001b[" + std::to_string(static_cast<uint8_t>(color.to4bits()) + 40) + "m\u001b[K";
00098       else
00099         return "\u001b[48;5;" + std::to_string(color.to8bits()) + "m\u001b[K";
00100     case Term::Terminfo::ColorMode::Bit24:
00101       if(color.getType() == Term::Color::Type::Bit3 || color.getType() == Term::Color::Type::Bit4)
      return "\u001b[" + std::to_string(static_cast<uint8_t>(color.to4bits()) + 40) + "m\u001b[K";
00102       else if(color.getType() == Term::Color::Type::Bit8)
00103         return "\u001b[48;5;" + std::to_string(color.to8bits()) + "m\u001b[K";
00104       else
00105         return "\u001b[48;2;" + std::to_string(color.to24bits()[0]) + ';' +
      std::to_string(color.to24bits()[1]) + ';' + std::to_string(color.to24bits()[2]) + "m\u001b[K";
00106     default: return {};
00107   }
00108 }
00109
00110 std::string Term::color_fg(const Term::Color::Name& name) { return color_fg(Color(name)); }
00111
00112 std::string Term::color_fg(const std::uint8_t& value) { return color_fg(Color(value)); }
00113
00114 std::string Term::color_fg(const std::uint8_t& red, const std::uint8_t& green, const std::uint8_t&
      blue) { return color_fg(Color(red, green, blue)); }
00115
00116 std::string Term::color_fg(const Color& color)
00117 {
00118   if(color.getType() == Term::Color::Type::Unset || color.getType() == Term::Color::Type::NoColor) {
      return {}; }
00119   switch(Term::Terminfo::getColorMode())
00120   {
00121     case Term::Terminfo::ColorMode::Unset:
00122     case Term::Terminfo::ColorMode::NoColor: return "";
00123     case Term::Terminfo::ColorMode::Bit3: return "\u001b[" +
      std::to_string(static_cast<uint8_t>(color.to3bits()) + 30) + "m";
00124     case Term::Terminfo::ColorMode::Bit4: return "\u001b[" +
      std::to_string(static_cast<uint8_t>(color.to4bits()) + 30) + "m";
00125     case Term::Terminfo::ColorMode::Bit8:
00126       if(color.getType() == Term::Color::Type::Bit4 || color.getType() == Term::Color::Type::Bit3)
      return "\u001b[" + std::to_string(static_cast<uint8_t>(color.to4bits()) + 30) + "m";
00127       else
00128         return "\u001b[38;5;" + std::to_string(color.to8bits()) + "m";
```

```
00129     case Term::Terminfo::ColorMode::Bit24:
00130       if(color.getType() == Term::Color::Type::Bit3 || color.getType() == Term::Color::Type::Bit4)
      return "\u001b[" + std::to_string(static_cast<uint8_t>(color.to4bits()) + 30) + "m";
00131       else if(color.getType() == Term::Color::Type::Bit8)
00132         return "\u001b[38;5;" + std::to_string(color.to8bits()) + "m";
00133       else
00134         return "\u001b[38;2;" + std::to_string(color.to24bits()[0]) + ';' +
      std::to_string(color.to24bits()[1]) + ';' + std::to_string(color.to24bits()[2]) + "m";
00135     default: return {};
00136   }
00137 }
```

9.9 cpp-terminal/color.hpp File Reference

```
#include "cpp-terminal/terminfo.hpp"
#include <array>
#include <cstdint>
#include <string>
```

Classes

- class Term::Color

Namespaces

- namespace Term

Functions

- std::string Term::color_bg (const Term::Color::Name &name)
- std::string Term::color_bg (const std::uint8_t &value)
- std::string Term::color_bg (const std::uint8_t &red, const std::uint8_t &green, const std::uint8_t &blue)
- std::string Term::color_bg (const Color &color)
- std::string Term::color_fg (const Term::Color::Name &name)
- std::string Term::color_fg (const std::uint8_t &value)
- std::string Term::color_fg (const std::uint8_t &red, const std::uint8_t &green, const std::uint8_t &blue)
- std::string Term::color_fg (const Color &color)

9.10 color.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/terminfo.hpp"
00013
00014 #include <array>
00015 #include <cstdint>
00016 #include <string>
00017
00018 namespace Term
```

```
00019 {
00020
00021 class Color
00022 {
00023 public:
00024   enum class Type : std::uint8_t
00025   {
00026     Unset,
00027     NoColor,
00028     Bit3,
00029     Bit4,
00030     Bit8,
00031     Bit24
00032   };
00033
00039   enum class Name : std::uint8_t
00040   {
00041     Black         = 0,
00042     Red           = 1,
00043     Green         = 2,
00044     Yellow        = 3,
00045     Blue          = 4,
00046     Magenta       = 5,
00047     Cyan          = 6,
00048     White         = 7,
00049     Default       = 9,
00050     Gray          = 60,
00051     BrightBlack   = 60,
00052     BrightRed     = 61,
00053     BrightGreen   = 62,
00054     BrightYellow  = 63,
00055     BrightBlue    = 64,
00056     BrightMagenta = 65,
00057     BrightCyan    = 66,
00058     BrightWhite   = 67
00059   };
00060   bool operator==(const Color&) const;
00061   bool operator!=(const Color&) const;
00062   Color();
00063   Color(const Term::Color::Name& name);
00064   Color(const std::uint8_t& color);
00065   Color(const std::uint8_t& red, const std::uint8_t& green, const std::uint8_t& blue);
00066   Type                        getType() const;
00067   Name                        to3bits() const;
00068   Name                        to4bits() const;
00069   std::uint8_t                to8bits() const;
00070   std::array<std::uint8_t, 3> to24bits() const;
00071
00072 private:
00073   Type m_Type{Type::Unset};
00074   union
00075   {
00076     std::uint8_t                m_bit8;
00077     std::array<std::uint8_t, 3> m_bit24;
00078   };
00079 };
00080
00081 std::string color_bg(const Term::Color::Name& name);
00082 std::string color_bg(const std::uint8_t& value);
00083 std::string color_bg(const std::uint8_t& red, const std::uint8_t& green, const std::uint8_t& blue);
00084 std::string color_bg(const Color& color);
00085
00086 std::string color_fg(const Term::Color::Name& name);
00087 std::string color_fg(const std::uint8_t& value);
00088 std::string color_fg(const std::uint8_t& red, const std::uint8_t& green, const std::uint8_t& blue);
00089 std::string color_fg(const Color& color);
00090
00091 }  // namespace Term
```

9.11 cpp-terminal/cursor.hpp File Reference

```
#include <cstddef>
#include <cstdint>
#include <string>
```

Classes

- class Term::Cursor

Namespaces

- namespace Term

Functions

- Term::Cursor Term::cursor_position ()
- std::string Term::cursor_move (const std::size_t &row, const std::size_t &column)
- std::string Term::cursor_up (const std::size_t &rows)
- std::string Term::cursor_down (const std::size_t &rows)
- std::string Term::cursor_left (const std::size_t &columns)
- std::string Term::cursor_right (const std::size_t &columns)
- std::string Term::cursor_position_report ()
- std::string Term::cursor_off ()
- std::string Term::cursor_on ()
- std::string Term::clear_eol ()

9.12 cursor.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstddef>
00013 #include <cstdint>
00014 #include <string>
00015
00016 namespace Term
00017 {
00018
00019 class Cursor
00020 {
00021 public:
00022   Cursor() = default;
00023   Cursor(const std::size_t& row, const std::size_t& column);
00024   std::size_t row() const;
00025   std::size_t column() const;
00026   void        setRow(const std::size_t&);
00027   void        setColum(const std::size_t&);
00028   bool        empty() const;
00029   bool        operator==(const Term::Cursor& cursor) const;
00030   bool        operator!=(const Term::Cursor& cursor) const;
00031
00032 private:
00033   std::pair<std::size_t, std::size_t> m_position;
00034 };
00035
00036 // returns the current cursor position (row, column) (Y, X)
00037 Term::Cursor cursor_position();
00038
00039 // move the cursor to the given (row, column) / (Y, X)
00040 std::string cursor_move(const std::size_t& row, const std::size_t& column);
00041 // move the cursor the given rows up
00042 std::string cursor_up(const std::size_t& rows);
00043 // move the cursor the given rows down
00044 std::string cursor_down(const std::size_t& rows);
00045 // move the cursor the given columns left
00046 std::string cursor_left(const std::size_t& columns);
00047 // move the cursor the given columns right
00048 std::string cursor_right(const std::size_t& columns);
00049 // the ANSI code to generate a cursor position report
00050 std::string cursor_position_report();
00051 // turn off the cursor
00052 std::string cursor_off();
```

```
00053 // turn on the cursor
00054 std::string cursor_on();
00055
00056 // clears the screen from the current cursor position to the end of the screen
00057 std::string clear_eol();
00058
00059 }  // namespace Term
```

9.13 cpp-terminal/event.cpp File Reference

```
#include "cpp-terminal/event.hpp"
#include "cpp-terminal/private/conversion.hpp"
#include <chrono>
```

9.14 event.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/event.hpp"
00011
00012 #include "cpp-terminal/private/conversion.hpp"
00013
00014 #include <chrono>
00015
00016 #if defined(_MSC_VER)
00017   // Disable stupid warnings on Windows
00018   #pragma warning(push)
00019   #pragma warning(disable : 4582)
00020   #pragma warning(disable : 4583)
00021 #endif
00022 Term::Event::container::container() {}
00023
00024 Term::Event::container::~container() {}
00025 #if defined(_MSC_VER)
00026   #pragma warning(pop)
00027 #endif
00028
00029 Term::Key* Term::Event::get_if_key()
00030 {
00031   if(m_Type == Type::Key) return &m_container.m_Key;
00032   return nullptr;
00033 }
00034
00035 const Term::Key* Term::Event::get_if_key() const
00036 {
00037   if(m_Type == Type::Key) return &m_container.m_Key;
00038   return nullptr;
00039 }
00040
00041 Term::Screen* Term::Event::get_if_screen()
00042 {
00043   if(m_Type == Type::Screen) return &m_container.m_Screen;
00044   return nullptr;
00045 }
00046
00047 Term::Mouse* Term::Event::get_if_mouse()
00048 {
00049   if(m_Type == Type::Mouse) return &m_container.m_Mouse;
00050   return nullptr;
00051 }
00052
00053 const Term::Mouse* Term::Event::get_if_mouse() const
00054 {
00055   if(m_Type == Type::Mouse) return &m_container.m_Mouse;
00056   return nullptr;
00057 }
00058
```

```
00059 const Term::Screen* Term::Event::get_if_screen() const
00060 {
00061   if(m_Type == Type::Screen) return &m_container.m_Screen;
00062   return nullptr;
00063 }
00064
00065 Term::Cursor* Term::Event::get_if_cursor()
00066 {
00067   if(m_Type == Type::Cursor) return &m_container.m_Cursor;
00068   return nullptr;
00069 }
00070
00071 const Term::Cursor* Term::Event::get_if_cursor() const
00072 {
00073   if(m_Type == Type::Cursor) return &m_container.m_Cursor;
00074   return nullptr;
00075 }
00076
00077 std::string* Term::Event::get_if_copy_paste()
00078 {
00079   if(m_Type == Type::CopyPaste) return &m_container.m_string;
00080   return nullptr;
00081 }
00082
00083 const std::string* Term::Event::get_if_copy_paste() const
00084 {
00085   if(m_Type == Type::CopyPaste) return &m_container.m_string;
00086   return nullptr;
00087 }
00088
00089 Term::Event::Event(const Term::Cursor& cursor) : m_Type(Type::Cursor) { m_container.m_Cursor = cursor;
      }
00090
00091 Term::Focus* Term::Event::get_if_focus()
00092 {
00093   if(m_Type == Type::Focus) return &m_container.m_Focus;
00094   return nullptr;
00095 }
00096
00097 const Term::Focus* Term::Event::get_if_focus() const
00098 {
00099   if(m_Type == Type::Focus) return &m_container.m_Focus;
00100   return nullptr;
00101 }
00102
00103 Term::Event& Term::Event::operator=(const Term::Event& event)
00104 {
00105   m_Type = event.m_Type;
00106   switch(m_Type)
00107   {
00108     case Type::Empty: break;
00109     case Type::Key: m_container.m_Key = event.m_container.m_Key; break;
00110     case Type::CopyPaste: new(&this->m_container.m_string) std::string(event.m_container.m_string);
      break;
00111     case Type::Cursor: m_container.m_Cursor = event.m_container.m_Cursor; break;
00112     case Type::Screen: m_container.m_Screen = event.m_container.m_Screen; break;
00113     case Type::Focus: m_container.m_Focus = event.m_container.m_Focus; break;
00114     case Type::Mouse: m_container.m_Mouse = event.m_container.m_Mouse; break;
00115     default: break;
00116   }
00117   return *this;
00118 }
00119
00120 Term::Event::Event(const Term::Focus& focus) : m_Type(Type::Focus) { m_container.m_Focus = focus; }
00121
00122 Term::Event::Event(const Term::Event& event)
00123 {
00124   m_Type = event.m_Type;
00125   switch(m_Type)
00126   {
00127     case Type::Empty: break;
00128     case Type::Key: m_container.m_Key = event.m_container.m_Key; break;
00129     case Type::CopyPaste: new(&this->m_container.m_string) std::string(event.m_container.m_string);
      break;
00130     case Type::Cursor: m_container.m_Cursor = event.m_container.m_Cursor; break;
00131     case Type::Screen: m_container.m_Screen = event.m_container.m_Screen; break;
00132     case Type::Focus: m_container.m_Focus = event.m_container.m_Focus; break;
00133     case Type::Mouse: m_container.m_Mouse = event.m_container.m_Mouse; break;
00134     default: break;
00135   }
00136 }
00137
00138 Term::Event::~Event()
00139 {
00140   using std::string;
00141   if(m_Type == Type::CopyPaste) { m_container.m_string.~string(); }
00142 }
```

```
00143
00144 Term::Event::Event() = default;
00145
00146 Term::Event::Event(Term::Event&& event) noexcept : m_Type(event.m_Type)
00147 {
00148   switch(m_Type)
00149   {
00150     case Type::Empty: break;
00151     case Type::Key: std::swap(m_container.m_Key, event.m_container.m_Key); break;
00152     case Type::CopyPaste: std::swap(m_container.m_string, event.m_container.m_string); break;
00153     case Type::Cursor: std::swap(m_container.m_Cursor, event.m_container.m_Cursor); break;
00154     case Type::Screen: std::swap(m_container.m_Screen, event.m_container.m_Screen); break;
00155     case Type::Focus: std::swap(m_container.m_Focus, event.m_container.m_Focus); break;
00156     case Type::Mouse: std::swap(m_container.m_Mouse, event.m_container.m_Mouse); break;
00157     default: break;
00158   }
00159 }
00160
00161 Term::Event& Term::Event::operator=(Term::Event&& other) noexcept
00162 {
00163   switch(other.m_Type)
00164   {
00165     case Type::Empty: break;
00166     case Type::Key: std::swap(m_container.m_Key, other.m_container.m_Key); break;
00167     case Type::CopyPaste: std::swap(m_container.m_string, other.m_container.m_string); break;
00168     case Type::Cursor: std::swap(m_container.m_Cursor, other.m_container.m_Cursor); break;
00169     case Type::Screen: std::swap(m_container.m_Screen, other.m_container.m_Screen); break;
00170     case Type::Focus: std::swap(m_container.m_Focus, other.m_container.m_Focus); break;
00171     case Type::Mouse: std::swap(m_container.m_Mouse, other.m_container.m_Mouse); break;
00172     default: break;
00173   }
00174   return *this;
00175 }
00176
00177 Term::Event::Event(const Term::Mouse& mouse) : m_Type(Type::Mouse) { m_container.m_Mouse = mouse; }
00178
00179 bool Term::Event::empty() const { return m_Type == Type::Empty; }
00180
00181 Term::Event::operator Term::Mouse() const
00182 {
00183   if(m_Type == Type::Mouse) { return m_container.m_Mouse; }
00184   return {};
00185 }
00186
00187 Term::Event::operator std::string() const
00188 {
00189   if(m_Type == Type::CopyPaste) { return m_container.m_string; }
00190   return {};
00191 }
00192
00193 Term::Event::operator Term::Screen() const
00194 {
00195   if(m_Type == Type::Screen) { return m_container.m_Screen; }
00196   return {};
00197 }
00198
00199 Term::Event::Event(const Term::Screen& screen) : m_Type(Type::Screen) { m_container.m_Screen = screen;
      }
00200
00201 Term::Event::Event(const Term::Key& key) : m_Type(Type::Key) { m_container.m_Key = key; }
00202
00203 Term::Event::Type Term::Event::type() const { return m_Type; }
00204
00205 Term::Event::Event(const std::string& str) { parse(str); }
00206
00207 void Term::Event::parse(const std::string& str)
00208 {
00209   if(str.empty()) m_Type = Type::Empty;
00210   else if(str.size() == 1)
00211   {
00212     m_Type            = Type::Key;
00213     m_container.m_Key = Key(static_cast<Term::Key>(str[0]));
00214     /* Backspace return 127 CTRL+backspace return 8 */
00215     if(m_container.m_Key == Term::Key::Del) m_container.m_Key = Key(Term::Key::Backspace);
00216   }
00217   else if(str == "\033[I")
00218   {
00219     m_Type              = Type::Focus;
00220     m_container.m_Focus = Term::Focus(Term::Focus::Type::In);
00221   }
00222   else if(str == "\033[O")
00223   {
00224     m_Type              = Type::Focus;
00225     m_container.m_Focus = Term::Focus(Term::Focus::Type::Out);
00226   }
00227   else if(str.size() == 2 && str[0] == '\033')
00228   {
```

```
00229     m_container.m_Key = Key(static_cast<Term::Key>(Term::MetaKey::Value::Alt +
      static_cast<Term::Key>(str[1])));
00230     m_Type            = Type::Key;
00231   }
00232   else if(str[0] == '\033' && str[1] == '[' && str[str.size() - 1] == 'R')
00233   {
00234     std::size_t found = str.find(';', 2);
00235     if(found != std::string::npos)
00236     {
00237       m_Type               = Type::Cursor;
00238       m_container.m_Cursor = Cursor(static_cast<std::uint16_t>(std::stoi(str.substr(2, found - 2))),
      static_cast<std::uint16_t>(std::stoi(str.substr(found + 1, str.size() - (found + 2)))));
00239     }
00240   }
00241   else if(str[0] == '\033' && str[1] == '[' && str[2] == '<')
00242   {
00243     static std::chrono::time_point<std::chrono::system_clock> old;
00244     bool                                                      not_too_long{false};
00245     if(std::chrono::system_clock::now() - old <= std::chrono::milliseconds{120}) not_too_long = true;
00246     m_Type = Type::Mouse;
00247     std::size_t              pos{3};
00248     std::size_t              pos2{3};
00249     std::vector<std::size_t> values;
00250     while((pos = str.find(';', pos)) != std::string::npos)
00251     {
00252       values.push_back(std::stoull(str.substr(pos2, pos - pos2)));
00253       pos++;
00254       pos2 = pos;
00255     }
00256     values.push_back(std::stoull(str.substr(pos2, str.size() - pos2 - 1)));
00257     static Term::Mouse   first;
00258     static Term::Mouse   second;
00259     Term::Button::Action action;
00260     if(str[str.size() - 1] == 'm') action = Term::Button::Action::Released;
00261     else
00262       action = Term::Button::Action::Pressed;
00263     Term::Button::Type type;
00264     switch(values[0])
00265     {
00266       case 0:
00267       {
00268         type = Term::Button::Type::Right;
00269         break;
00270       }
00271       case 1:
00272       {
00273         type = Term::Button::Type::Wheel;
00274         break;
00275       }
00276       case 2:
00277       {
00278         type = Term::Button::Type::Left;
00279         break;
00280       }
00281       case 35:
00282       {
00283         type   = Term::Button::Type::None;
00284         action = Term::Button::Action::None;
00285         break;
00286       }
00287       case 64:
00288       {
00289         type   = Term::Button::Type::Wheel;
00290         action = Term::Button::Action::RolledUp;
00291         break;
00292       }
00293       case 65:
00294       {
00295         type   = Term::Button::Type::Wheel;
00296         action = Term::Button::Action::RolledDown;
00297         break;
00298       }
00299       default: break;
00300     }
00301     if(not_too_long && first.row() == second.row() && second.row() == values[1] && first.column() ==
      second.column() && second.column() == values[2] && first.getButton().type() ==
      second.getButton().type() && second.getButton().type() == type && first.getButton().action() ==
      Button::Action::Released && second.getButton().action() == Button::Action::Pressed && action ==
      Button::Action::Pressed) action = Term::Button::Action::DoubleClicked;
00302     second              = first;
00303     first               = Term::Mouse(Term::Button(type, action), values[1], values[2]);
00304     m_container.m_Mouse = first;
00305     old                 = std::chrono::system_clock::now();
00306   }
00307   else if(str.size() <= 10)
00308   {
00309     //https://invisible-island.net/xterm/ctlseqs/ctlseqs.html
```

```
00310     // CSI = ESC[ SS3 = ESCO
00311     /*
00312      * Key            Normal     Application
00313      * -------------+----------+-------------
00314      * Cursor Up    | CSI A    | SS3 A
00315      * Cursor Down  | CSI B    | SS3 B
00316      * Cursor Right | CSI C    | SS3 C
00317      * Cursor Left  | CSI D    | SS3 D
00318      * -------------+----------+-------------
00319      * Key            Normal/Application
00320      * -------------+--------------------
00321      * Cursor Up    | ESC A
00322      * Cursor Down  | ESC B
00323      * Cursor Right | ESC C
00324      * Cursor Left  | ESC D
00325      * -------------+--------------------
00326     */
00327     if((str == "\u001bOA") || (str == "\u001b[A") || (str == "\u001bA")) { m_container.m_Key =
      Key(Term::Key::ArrowUp); }
00328     else if((str == "\u001bOB") || (str == "\u001b[B") || (str == "\u001bB")) { m_container.m_Key =
      Key(Term::Key::ArrowDown); }
00329     else if((str == "\u001bOC") || (str == "\u001b[C") || (str == "\u001bC")) { m_container.m_Key =
      Key(Term::Key::ArrowRight); }
00330     else if((str == "\u001bOD") || (str == "\u001b[D") || (str == "\u001bD")) { m_container.m_Key =
      Key(Term::Key::ArrowLeft); }
00331     /*
00332      * Key        Normal     Application
00333      * ---------+----------+-------------
00334      * Home     | CSI H    | SS3 H
00335      * End      | CSI F    | SS3 F
00336      * ---------+----------+-------------
00337     */
00338     else if((str == "\u001bOH") || (str == "\u001b[H"))
00339       m_container.m_Key = Key(Term::Key::Home);
00340     else if(str == "\u001bOF" || str == "\u001b[F")
00341       m_container.m_Key = Key(Term::Key::End);
00342     /*
00343      * Key        Escape Sequence
00344      * ---------+-----------------
00345      * F1       | SS3 P
00346      * F2       | SS3 Q
00347      * F3       | SS3 R
00348      * F4       | SS3 S
00349      * F1       | CSI 1 1 ~
00350      * F2       | CSI 1 2 ~
00351      * F3       | CSI 1 3 ~
00352      * F4       | CSI 1 4 ~
00353      * F5       | CSI 1 5 ~
00354      * F6       | CSI 1 7 ~
00355      * F7       | CSI 1 8 ~
00356      * F8       | CSI 1 9 ~
00357      * F9       | CSI 2 0 ~
00358      * F10      | CSI 2 1 ~
00359      * F11      | CSI 2 3 ~
00360      * F12      | CSI 2 4 ~
00361      * ---------+-----------------
00362     */
00363     else if(str == "\u001bOP" || str == "\u001b[11~")
00364       m_container.m_Key = Key(Term::Key::F1);
00365     else if(str == "\u001bOQ" || str == "\u001b[12~")
00366       m_container.m_Key = Key(Term::Key::F2);
00367     else if(str == "\u001bOR" || str == "\u001b[13~")
00368       m_container.m_Key = Key(Term::Key::F3);
00369     else if(str == "\u001bOS" || str == "\u001b[14~")
00370       m_container.m_Key = Key(Term::Key::F4);
00371     else if(str == "\u001b[15~")
00372       m_container.m_Key = Key(Term::Key::F5);
00373     else if(str == "\u001b[17~")
00374       m_container.m_Key = Key(Term::Key::F6);
00375     else if(str == "\u001b[18~")
00376       m_container.m_Key = Key(Term::Key::F7);
00377     else if(str == "\u001b[19~")
00378       m_container.m_Key = Key(Term::Key::F8);
00379     else if(str == "\u001b[20~")
00380       m_container.m_Key = Key(Term::Key::F9);
00381     else if(str == "\u001b[21~")
00382       m_container.m_Key = Key(Term::Key::F10);
00383     else if(str == "\u001b[23~")
00384       m_container.m_Key = Key(Term::Key::F11);
00385     else if(str == "\u001b[24~")
00386       m_container.m_Key = Key(Term::Key::F12);
00387     /*
00388      * Key        Normal     Application
00389      * ---------+----------+-------------
00390      * Insert   | CSI 2 ~  | CSI 2 ~
00391      * Delete   | CSI 3 ~  | CSI 3 ~
00392      * Home     | CSI 1 ~  | CSI 1 ~
```

```
00393       * End      | CSI 4 ~  | CSI 4 ~
00394       * PageUp   | CSI 5 ~  | CSI 5 ~
00395       * PageDown | CSI 6 ~  | CSI 6 ~
00396       * ---------+----------+-------------
00397       */
00398      else if(str == "\u001b[2~") { m_container.m_Key = Key(Term::Key::Insert); }
00399      else if(str == "\u001b[3~") { m_container.m_Key = Key(Term::Key::Del); }
00400      else if(str == "\u001b[1~") { m_container.m_Key = Key(Term::Key::Home); }
00401      else if(str == "\u001b[4~") { m_container.m_Key = Key(Term::Key::End); }
00402      else if(str == "\u001b[5~") { m_container.m_Key = Key(Term::Key::PageUp); }
00403      else if(str == "\u001b[6~") { m_container.m_Key = Key(Term::Key::PageDown); }
00404      /*
00405       * Key        Escape Sequence
00406       * ---------+-----------------
00407       * F13      | CSI 2 5 ~
00408       * F14      | CSI 2 6 ~
00409       * F15      | CSI 2 8 ~
00410       * F16      | CSI 2 9 ~
00411       * F17      | CSI 3 1 ~
00412       * F18      | CSI 3 2 ~
00413       * F19      | CSI 3 3 ~
00414       * F20      | CSI 3 4 ~
00415       * ---------+-----------------
00416       */
00417      else if(str == "\u001b[25~")
00418        m_container.m_Key = Key(Term::Key::F13);
00419      else if(str == "\u001b[26~")
00420        m_container.m_Key = Key(Term::Key::F14);
00421      else if(str == "\u001b[28~")
00422        m_container.m_Key = Key(Term::Key::F15);
00423      else if(str == "\u001b[29~")
00424        m_container.m_Key = Key(Term::Key::F16);
00425      else if(str == "\u001b[31~")
00426        m_container.m_Key = Key(Term::Key::F17);
00427      else if(str == "\u001b[32~")
00428        m_container.m_Key = Key(Term::Key::F18);
00429      else if(str == "\u001b[33~")
00430        m_container.m_Key = Key(Term::Key::F19);
00431      else if(str == "\u001b[34~")
00432        m_container.m_Key = Key(Term::Key::F20);
00433      else if(str == "\u001b[G")
00434        m_container.m_Key = Key(Term::Key::Value::Numeric5);
00435      else if(Term::Private::is_valid_utf8_code_unit(str))
00436        m_container.m_Key = Key(static_cast<Term::Key::Value>(Term::Private::utf8_to_utf32(str)[0]));
00437      else
00438      {
00439        m_Type = Type::CopyPaste;
00440        new(&this->m_container.m_string) std::string(str);
00441        return;
00442      }
00443      m_Type = Type::Key;
00444    }
00445    else
00446    {
00447      m_Type = Type::CopyPaste;
00448      new(&this->m_container.m_string) std::string(str);
00449    }
00450  }
00451
00452  Term::Event::operator Term::Key() const
00453  {
00454    if(m_Type == Type::Key) { return m_container.m_Key; }
00455    return {};
00456  }
00457
00458  Term::Event::operator Term::Cursor() const
00459  {
00460    if(m_Type == Type::Cursor) { return m_container.m_Cursor; }
00461    return {};
00462  }
00463
00464  Term::Event::operator Term::Focus() const
00465  {
00466    if(m_Type == Type::Focus) { return m_container.m_Focus; }
00467    return {};
00468  }
```

9.15 cpp-terminal/event.hpp File Reference

```
#include "cpp-terminal/cursor.hpp"
#include "cpp-terminal/focus.hpp"
```

```
#include "cpp-terminal/key.hpp"
#include "cpp-terminal/mouse.hpp"
#include "cpp-terminal/screen.hpp"
#include <cstdint>
#include <string>
```

Classes

- class Term::Event
- union Term::Event::container

Namespaces

- namespace Term

9.16 event.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/cursor.hpp"
00013 #include "cpp-terminal/focus.hpp"
00014 #include "cpp-terminal/key.hpp"
00015 #include "cpp-terminal/mouse.hpp"
00016 #include "cpp-terminal/screen.hpp"
00017
00018 #include <cstdint>
00019 #include <string>
00020
00021 namespace Term
00022 {
00023
00024 class Event
00025 {
00026 public:
00027   enum class Type
00028   {
00029     Empty,
00030     Key,
00031     Screen,
00032     Cursor,
00033     Focus,
00034     Mouse,
00035     CopyPaste,
00036   };
00037   ~Event();
00038   Event();
00039   Event(const std::string& str);
00040   Event(const Term::Key& key);
00041   Event(const Term::Screen& screen);
00042   Event(const Term::Cursor& cursor);
00043   Event(const Term::Focus& focus);
00044   Event(const Term::Mouse& mouse);
00045   Event(const Term::Event& event);
00046   Event(Term::Event&& event) noexcept;
00047   Event& operator=(Event&& other) noexcept;
00048   Event& operator=(const Term::Event& event);
00049   bool   empty() const;
00050   Type   type() const;
00051   operator Term::Key() const;
00052   operator Term::Screen() const;
00053   operator Term::Cursor() const;
```

```
00054   operator Term::Focus() const;
00055   operator Term::Mouse() const;
00056   operator std::string() const;
00057
00058   // getters
00059   Key*               get_if_key();
00060   const Key*         get_if_key() const;
00061   Screen*            get_if_screen();
00062   const Screen*      get_if_screen() const;
00063   Cursor*            get_if_cursor();
00064   const Cursor*      get_if_cursor() const;
00065   Focus*             get_if_focus();
00066   const Focus*       get_if_focus() const;
00067   Mouse*             get_if_mouse();
00068   const Mouse*       get_if_mouse() const;
00069   std::string*       get_if_copy_paste();
00070   const std::string* get_if_copy_paste() const;
00071
00072 private:
00073   void parse(const std::string& str);
00074   union container
00075   {
00076     container();
00077     ~container();
00078     container(const container&)              = delete;
00079     container(container&&)                   = delete;
00080     container&   operator=(const container&) = delete;
00081     container&   operator=(container&&)      = delete;
00082     Term::Key    m_Key;
00083     Term::Cursor m_Cursor;
00084     Term::Screen m_Screen;
00085     Term::Focus  m_Focus;
00086     Term::Mouse  m_Mouse;
00087     std::string  m_string;
00088   };
00089   Type      m_Type{Type::Empty};
00090   container m_container;
00091 };
00092
00093 }  // namespace Term
```

9.17 cpp-terminal/exception.hpp File Reference

```
#include <cstdint>
#include <exception>
#include <string>
```

Classes

- class Term::Exception

Namespaces

- namespace Term

9.18 exception.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
```

```
00010 #pragma once
00011
00012 #include <cstdint>
00013 #include <exception>
00014 #include <string>
00015
00016 namespace Term
00017 {
00018
00019 class Exception : public std::exception
00020 {
00021 public:
00022   explicit Exception(const std::string& message) noexcept;
00023   Exception(const std::int64_t& code, const std::string& message) noexcept;
00024   Exception(const Exception&)            = default;
00025   Exception(Exception&&)                 = default;
00026   Exception& operator=(Exception&&)      = default;
00027   Exception& operator=(const Exception&) = default;
00028
00029   const char*  what() const noexcept override;
00030   std::int64_t code() const noexcept;
00031   std::string  message() const noexcept;
00032   std::string  context() const noexcept;
00033   ~Exception() noexcept override = default;
00034
00035 protected:
00036   explicit Exception(const std::int64_t& code) noexcept;
00037   virtual void                      build_what() const noexcept;
00038   void                              setMessage(const std::string& message) noexcept;
00039   void                              setContext(const std::string& context) noexcept;
00040   void                              setWhat(const std::string& what) const noexcept;
00041   static const constexpr std::size_t m_maxSize{256};
00042
00043 private:
00044   std::int64_t        m_code{0};
00045   std::string         m_message;
00046   std::string         m_context;
00047   mutable std::string m_what;
00048 };
00049
00050 }  // namespace Term
```

9.19 cpp-terminal/private/exception.hpp File Reference

```
#include "cpp-terminal/exception.hpp"
#include <cstdint>
#include <string>
```

Classes

- class Term::Private::WindowsError
- class Term::Private::WindowsException
- class Term::Private::Errno
- class Term::Private::ErrnoException

Namespaces

- namespace Term
- namespace Term::Private

Enumerations

- enum class Term::Private::ExceptionDestination : std::uint8_t { Term::Private::MessageBox = 0 , Term::Private::StdErr }

Functions

- void Term::Private::ExceptionHandler (const ExceptionDestination &destination=ExceptionDestination::StdErr) noexcept

9.20 exception.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/exception.hpp"
00013
00014 #include <cstdint>
00015 #include <string>
00016
00017 namespace Term
00018 {
00019
00020 namespace Private
00021 {
00022
00023 #if defined(_WIN32)
00024 class WindowsError
00025 {
00026 public:
00027   WindowsError(const WindowsError&)                    = default;
00028   WindowsError(WindowsError&&)                         = default;
00029   WindowsError() noexcept                              = default;
00030   virtual ~WindowsError() noexcept                     = default;
00031   WindowsError& operator=(WindowsError&&) noexcept     = default;
00032   WindowsError& operator=(const WindowsError&) noexcept = default;
00033   std::int64_t  error() const noexcept;
00034   bool          check_value() const noexcept;
00035   WindowsError& check_if(const bool& ret) noexcept;
00036   void          throw_exception(const std::string& str = std::string()) const;
00037
00038 private:
00039   std::int64_t m_error{0};
00040   bool         m_check_value{false};
00041 };
00042
00043 class WindowsException : public Term::Exception
00044 {
00045 public:
00046   WindowsException(const std::int64_t& error, const std::string& context = std::string());
00047   ~WindowsException() override = default;
00048
00049 private:
00050   void build_what() const noexcept final;
00051 };
00052 #endif
00053
00054 class Errno
00055 {
00056 public:
00057   Errno(const Errno&) noexcept = default;
00058   Errno(Errno&&) noexcept      = default;
00059   Errno() noexcept;
00060   virtual ~Errno() noexcept;
00061   Errno&       operator=(Errno&&) noexcept      = default;
00062   Errno&       operator=(const Errno&) noexcept = default;
00063   std::int64_t error() const noexcept;
00064   bool         check_value() const noexcept;
00065   Errno&       check_if(const bool& ret) noexcept;
00066   void         throw_exception(const std::string& str = {}) const;
00067
00068 private:
00069   std::int64_t m_errno{0};
00070   bool         m_check_value{false};
00071 };
00072
00073 class ErrnoException : public Term::Exception
```

```
00074 {
00075 public:
00076   ErrnoException(const ErrnoException&) = default;
00077   ErrnoException(ErrnoException&&)      = default;
00078   explicit ErrnoException(const std::int64_t& error, const std::string& context = {});
00079   ~ErrnoException() override                       = default;
00080   ErrnoException& operator=(ErrnoException&&)      = default;
00081   ErrnoException& operator=(const ErrnoException&) = default;
00082
00083 private:
00084   void build_what() const noexcept final;
00085 };
00086
00087 enum class ExceptionDestination : std::uint8_t
00088 {
00089   MessageBox = 0,
00090   StdErr,
00091 };
00092
00093 void ExceptionHandler(const ExceptionDestination& destination = ExceptionDestination::StdErr)
      noexcept;
00094
00095 }  // namespace Private
00096
00097 }  // namespace Term
```

9.21 cpp-terminal/focus.cpp File Reference

```
#include "cpp-terminal/focus.hpp"
```

Namespaces

- namespace Term

9.22 focus.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/focus.hpp"
00011
00012 namespace Term
00013 {
00014
00015 Focus::Focus(const Term::Focus::Type& type) : m_focus(type) {}
00016
00017 Term::Focus::Type Focus::type() const { return m_focus; }
00018
00019 bool Focus::in() const { return m_focus == Term::Focus::Type::In; }
00020
00021 bool Focus::out() const { return m_focus == Term::Focus::Type::Out; }
00022
00023 bool Term::Focus::operator==(const Term::Focus& focus) const { return m_focus == focus.m_focus; }
00024
00025 bool Term::Focus::operator!=(const Term::Focus& focus) const { return !(*this == focus); }
00026
00027 }  // namespace Term
```

9.23 cpp-terminal/focus.hpp File Reference

```
#include <cstdint>
```

Classes

- class Term::Focus

  Class to return the focus of the terminal.

Namespaces

- namespace Term

9.24 focus.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstdint>
00013
00014 namespace Term
00015 {
00016
00021 class Focus
00022 {
00023 public:
00024   enum class Type : std::int8_t
00025   {
00026     Unknown = -1,
00027     Out     = 0,
00028     In      = 1,
00029   };
00030
00031   Focus() = default;
00032
00033   explicit Focus(const Term::Focus::Type& type);
00034
00040   Term::Focus::Type type() const;
00041
00048   bool in() const;
00049
00056   bool out() const;
00057
00058   bool operator==(const Term::Focus& focus) const;
00059   bool operator!=(const Term::Focus& focus) const;
00060
00061 private:
00062   Term::Focus::Type m_focus{Term::Focus::Type::Unknown};
00063 };
00064
00065 }  // namespace Term
```

9.25 cpp-terminal/input.hpp File Reference

```
#include "cpp-terminal/event.hpp"
```

Namespaces

- namespace Term

Functions

- Term::Event Term::read_event ()

9.26 input.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/event.hpp"
00013
00014 namespace Term
00015 {
00016
00017 Term::Event read_event();
00018
00019 }  // namespace Term
```

9.27 cpp-terminal/private/input.hpp File Reference

```
#include "cpp-terminal/event.hpp"
#include <cstdint>
#include <thread>
```

Classes

- class Term::Private::Input

Namespaces

- namespace Term
- namespace Term::Private

9.28 input.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/event.hpp"
00013
00014 #include <cstdint>
00015 #include <thread>
00016
```

```
00017 namespace Term
00018 {
00019
00020 namespace Private
00021 {
00022
00023 class BlockingQueue;
00024
00025 class Input
00026 {
00027 public:
00028   Input();
00029   static void        startReading();
00030   static Term::Event getEvent();
00031   static Term::Event getEventBlocking();
00032
00033 private:
00034   static void read_event();
00035   static void read_raw();
00036 #if defined(_WIN32)
00037   static void read_windows_key(const std::uint16_t& virtual_key_code, const std::uint32_t&
      control_key_state, const std::size_t& occurrence);
00038 #endif
00039   static void                          init_thread();
00040   static std::thread                   m_thread;
00041   static Term::Private::BlockingQueue m_events;
00042   static int                           m_poll;  // for linux
00043 };
00044
00045 }  // namespace Private
00046
00047 }  // namespace Term
```

9.29 cpp-terminal/iostream.cpp File Reference

```
#include "cpp-terminal/iostream.hpp"
#include <array>
```

9.30 iostream.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/iostream.hpp"
00011
00012 #include <array>
00013
00014 namespace
00015 {
00016 std::array<char, sizeof(Term::TOstream)> cout_buffer;
      //NOLINT(fuchsia-statically-constructed-objects)
00017 std::array<char, sizeof(Term::TOstream)> clog_buffer;
      //NOLINT(fuchsia-statically-constructed-objects)
00018 std::array<char, sizeof(Term::TOstream)> cerr_buffer;
      //NOLINT(fuchsia-statically-constructed-objects)
00019 std::array<char, sizeof(Term::TIstream)> cin_buffer;
      //NOLINT(fuchsia-statically-constructed-objects)
00020 }  // namespace
00021
00022 Term::TOstream& Term::cout = reinterpret_cast<Term::TOstream&>(cout_buffer);
      //NOLINT(cppcoreguidelines-pro-type-reinterpret-cast)
00023 Term::TOstream& Term::clog = reinterpret_cast<Term::TOstream&>(clog_buffer);
      //NOLINT(cppcoreguidelines-pro-type-reinterpret-cast)
00024 Term::TOstream& Term::cerr = reinterpret_cast<Term::TOstream&>(cerr_buffer);
      //NOLINT(cppcoreguidelines-pro-type-reinterpret-cast)
00025 Term::TIstream& Term::cin  = reinterpret_cast<Term::TIstream&>(cin_buffer);
      //NOLINT(cppcoreguidelines-pro-type-reinterpret-cast)
```

9.31 cpp-terminal/iostream.hpp File Reference

```
#include "cpp-terminal/iostream_initializer.hpp"
#include "cpp-terminal/stream.hpp"
```

Namespaces

- namespace Term

Variables

- TIstream & Term::cin = reinterpret_cast<Term::TIstream&>(cin_buffer)
- TOstream & Term::cout = reinterpret_cast<Term::TOstream&>(cout_buffer)
- TOstream & Term::cerr = reinterpret_cast<Term::TOstream&>(cerr_buffer)
- TOstream & Term::clog = reinterpret_cast<Term::TOstream&>(clog_buffer)

9.32 iostream.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/iostream_initializer.hpp"
00013 #include "cpp-terminal/stream.hpp"
00014
00015 namespace Term
00016 {
00017
00018 extern TIstream& cin;
00019 extern TOstream& cout;
00020 extern TOstream& cerr;
00021 extern TOstream& clog;
00022
00023 static const IOStreamInitializer IO_stream_initializer;
      //NOLINT(cert-err58-cpp,fuchsia-statically-constructed-objects)
00024
00025 }  // namespace Term
```

9.33 cpp-terminal/iostream_initializer.cpp File Reference

```
#include "cpp-terminal/iostream_initializer.hpp"
#include "cpp-terminal/iostream.hpp"
#include "cpp-terminal/private/exception.hpp"
#include "cpp-terminal/terminal.hpp"
#include "cpp-terminal/terminal_initializer.hpp"
#include "cpp-terminal/tty.hpp"
#include <cstddef>
#include <iostream>
```

9.34 iostream_initializer.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/iostream_initializer.hpp"
00011
00012 #include "cpp-terminal/iostream.hpp"
00013 #include "cpp-terminal/private/exception.hpp"
00014 #include "cpp-terminal/terminal.hpp"
00015 #include "cpp-terminal/terminal_initializer.hpp"
00016 #include "cpp-terminal/tty.hpp"
00017
00018 #include <cstddef>
00019 #include <iostream>
00020
00021 std::size_t Term::IOStreamInitializer::m_counter{0};
00022
00023 Term::IOStreamInitializer::IOStreamInitializer() noexcept
00024 try
00025 {
00026   if(0 == m_counter)
00027   {
00028     static const std::ios_base::Init       iostreams_init;  // Init std::cout etc...
00029     static const Term::TerminalInitializer terminal_init;   // Make sure terminal is set up.
00030     new(&Term::cout) TOstream(Term::Buffer::Type::FullBuffered, BUFSIZ);
00031     new(&Term::clog) TOstream(Term::Buffer::Type::LineBuffered, BUFSIZ);
00032     new(&Term::cerr) TOstream(Term::Buffer::Type::Unbuffered, 0);
00033     new(&Term::cin) TIstream(Term::Buffer::Type::FullBuffered, BUFSIZ);
00034     if(is_stdin_a_tty()) { std::cin.rdbuf(Term::cin.rdbuf()); }
00035   }
00036   ++m_counter;
00037 }
00038 catch(...)
00039 {
00040   ExceptionHandler(Private::ExceptionDestination::StdErr);
00041 }
00042
00043 Term::IOStreamInitializer::~IOStreamInitializer() noexcept
00044 try
00045 {
00046   --m_counter;
00047   if(0 == m_counter)
00048   {
00049     (&Term::cout)->~TOstream();
00050     (&Term::cerr)->~TOstream();
00051     (&Term::clog)->~TOstream();
00052     (&Term::cin)->~TIstream();
00053   }
00054 }
00055 catch(...)
00056 {
00057   ExceptionHandler(Private::ExceptionDestination::StdErr);
00058 }
```

9.35 cpp-terminal/iostream_initializer.hpp File Reference

`#include <cstddef>`

Classes

- class Term::IOStreamInitializer

Namespaces

- namespace Term

9.36 iostream_initializer.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstddef>
00013
00014 namespace Term
00015 {
00016
00017 class IOStreamInitializer
00018 {
00019 public:
00020   ~IOStreamInitializer() noexcept;
00021   IOStreamInitializer() noexcept;
00022   IOStreamInitializer(const IOStreamInitializer&)            = delete;
00023   IOStreamInitializer(IOStreamInitializer&&)                 = delete;
00024   IOStreamInitializer& operator=(IOStreamInitializer&&)      = delete;
00025   IOStreamInitializer& operator=(const IOStreamInitializer&) = delete;
00026
00027 private:
00028   static std::size_t m_counter;
00029 };
00030
00031 }  // namespace Term
```

9.37 cpp-terminal/key.cpp File Reference

```
#include "cpp-terminal/key.hpp"
#include "cpp-terminal/private/unicode.hpp"
```

9.38 key.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/key.hpp"
00011
00012 #include "cpp-terminal/private/unicode.hpp"
00013
00014 // ------------------------- Key ----------------------------
00015
00016 void Term::Key::append_name(std::string& strOut) const
00017 {
00018   Term::Key key = *this;
00019   if(key == Term::Key::NoKey) return;
00020   if(key.hasAlt())
00021   {
00022     strOut += "Alt+";
00023     key = static_cast<Term::Key>(key.value - static_cast<std::int32_t>(Term::MetaKey::Value::Alt));
00024   }
00025   if(key.hasCtrl())
00026   {
00027     strOut += "Ctrl+";
00028     if(!key.iscntrl()) key = static_cast<Term::Key>(key.value -
      static_cast<std::int32_t>(Term::MetaKey::Value::Ctrl));
00029   }
00030   if(key == Term::Key::Tab) strOut += "Tab";
```

```
00031   else if(key == Term::Key::Enter)
00032     strOut += "Enter";
00033   else if(key == Term::Key::Esc)
00034     strOut += "Esc";
00035   else if(key == Term::Key::Backspace)
00036     strOut += "Backspace";
00037   else if(key == Term::Key::Del)
00038     strOut += "Del";
00039   else if(key.iscntrl())
00040     strOut += static_cast<char>(key.value + 64);
00041   else if(key == Term::Key::Space)
00042     strOut += "Space";
00043   else if(key.isunicode()) { strOut +=
      Term::Private::utf32_to_utf8(static_cast<char32_t>(this->value)); }
00044   else
00045   {
00046     switch(key)
00047     {
00048       case Term::Key::ArrowLeft: strOut += "Left Arrow"; break;
00049       case Term::Key::ArrowRight: strOut += "Right Arrow"; break;
00050       case Term::Key::ArrowUp: strOut += "Up arrow"; break;
00051       case Term::Key::ArrowDown: strOut += "Down arrow"; break;
00052       case Term::Key::Numeric5: strOut += "5 Numeric pad"; break;
00053       case Term::Key::Home: strOut += "Home"; break;
00054       case Term::Key::Insert: strOut += "Insert"; break;
00055       case Term::Key::End: strOut += "End"; break;
00056       case Term::Key::PageUp: strOut += "Page up"; break;
00057       case Term::Key::PageDown: strOut += "Page down"; break;
00058       case Term::Key::F1: strOut += "F1"; break;
00059       case Term::Key::F2: strOut += "F2"; break;
00060       case Term::Key::F3: strOut += "F3"; break;
00061       case Term::Key::F4: strOut += "F4"; break;
00062       case Term::Key::F5: strOut += "F5"; break;
00063       case Term::Key::F6: strOut += "F6"; break;
00064       case Term::Key::F7: strOut += "F7"; break;
00065       case Term::Key::F8: strOut += "F8"; break;
00066       case Term::Key::F9: strOut += "F9"; break;
00067       case Term::Key::F10: strOut += "F10"; break;
00068       case Term::Key::F11: strOut += "F11"; break;
00069       case Term::Key::F12: strOut += "F12"; break;
00070       case Term::Key::F13: strOut += "F13"; break;
00071       case Term::Key::F14: strOut += "F14"; break;
00072       case Term::Key::F15: strOut += "F15"; break;
00073       case Term::Key::F16: strOut += "F16"; break;
00074       case Term::Key::F17: strOut += "F17"; break;
00075       case Term::Key::F18: strOut += "F18"; break;
00076       case Term::Key::F19: strOut += "F19"; break;
00077       case Term::Key::F20: strOut += "F20"; break;
00078       case Term::Key::F21: strOut += "F21"; break;
00079       case Term::Key::F22: strOut += "F22"; break;
00080       case Term::Key::F23: strOut += "F23"; break;
00081       case Term::Key::F24: strOut += "F24"; break;
00082       case Term::Key::PrintScreen: strOut += "Print Screen"; break;
00083       case Term::Key::Menu: strOut += "Menu"; break;
00084       default: break;
00085     }
00086   }
00087 }
00088
00089 std::string Term::Key::name() const
00090 {
00091   std::string str;
00092   this->append_name(str);
00093   return str;
00094 }
00095
00096 std::string Term::Key::str() const { return
      Term::Private::utf32_to_utf8(static_cast<char32_t>(this->value)); }
```

9.39 cpp-terminal/key.hpp File Reference

```
#include <cstdint>
#include <string>
```

Classes

- class Term::MetaKey
- class Term::Key

Namespaces

- namespace Term

Functions

- constexpr bool Term::operator== (Key l, MetaKey r)
- constexpr bool Term::operator== (MetaKey l, Key r)
- constexpr bool Term::operator< (MetaKey l, Key r)
- constexpr bool Term::operator< (Key l, MetaKey r)
- constexpr bool Term::operator!= (Key l, MetaKey r)
- constexpr bool Term::operator!= (MetaKey l, Key r)
- constexpr bool Term::operator>= (MetaKey l, Key r)
- constexpr bool Term::operator>= (Key l, MetaKey r)
- constexpr bool Term::operator> (MetaKey l, Key r)
- constexpr bool Term::operator> (Key l, MetaKey r)
- constexpr bool Term::operator<= (MetaKey l, Key r)
- constexpr bool Term::operator<= (Key l, MetaKey r)
- constexpr Key Term::operator+ (MetaKey metakey, Key key)
- constexpr Key Term::operator+ (Key key, MetaKey meta)
- constexpr Key Term::operator+ (MetaKey::Value l, Key r)
- constexpr Key Term::operator+ (Key l, MetaKey::Value r)
- constexpr Key Term::operator+ (MetaKey::Value l, Key::value_type r)
- constexpr Key Term::operator+ (Key::value_type l, MetaKey::Value r)

9.40 key.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstdint>
00013 #include <string>
00014
00015 namespace Term
00016 {
00017
00018 class MetaKey
00019 {
00020 public:
00021   enum class Value : std::int32_t
00022   {
00023     // Last utf8 codepoint is U+10FFFF (000100001111111111111111) So:
00024     None = 0,
00025     Alt  = (1UL << 22UL),
00026     Ctrl = (1UL << 23UL),
00027   };
00028
00029   constexpr MetaKey() : value(static_cast<std::int32_t>(Value::None)) {}
00030   constexpr MetaKey(const MetaKey& key)          = default;
00031   inline MetaKey& operator=(const MetaKey& key) = default;
00032
00033   constexpr MetaKey(const Value& v) : value(static_cast<std::int32_t>(v)) {}
00034   inline MetaKey& operator=(const Value& v)
00035   {
00036     this->value = static_cast<std::int32_t>(v);
00037     return *this;
00038   }
00039
```

```
00040   explicit constexpr MetaKey(std::int32_t val) : value(val) {}
00041   inline MetaKey& operator=(std::int32_t val)
00042   {
00043     this->value = val;
00044     return *this;
00045   }
00046
00047   explicit constexpr operator std::int32_t() const { return this->value; }
00048
00049   constexpr bool hasAlt() const { return (this->value &
      static_cast<std::int32_t>(MetaKey::Value::Alt)) == static_cast<std::int32_t>(MetaKey::Value::Alt); }
00050   constexpr bool hasCtrl() const { return (this->value &
      static_cast<std::int32_t>(MetaKey::Value::Ctrl)) == static_cast<std::int32_t>(MetaKey::Value::Ctrl); }
00051
00052   friend constexpr MetaKey operator+(MetaKey l, MetaKey r) { return MetaKey(l.value | r.value); }
00053   friend constexpr MetaKey operator+(MetaKey::Value l, MetaKey::Value r) { return MetaKey(l) +
      MetaKey(r); }
00054   friend constexpr MetaKey operator+(MetaKey l, MetaKey::Value r) { return l + MetaKey(r); }
00055   friend constexpr MetaKey operator+(MetaKey::Value l, MetaKey r) { return MetaKey(l) + r; }
00056
00057   MetaKey& operator+=(MetaKey r) { return *this = *this + r; }
00058   MetaKey& operator+=(MetaKey::Value r) { return *this = *this + r; }
00059
00060   friend constexpr bool operator==(MetaKey l, MetaKey r) { return l.value == r.value; }
00061   friend constexpr bool operator==(MetaKey l, MetaKey::Value r) { return l == MetaKey(r); }
00062   friend constexpr bool operator==(MetaKey::Value l, MetaKey r) { return MetaKey(l) == r; }
00063
00064   friend constexpr bool operator!=(MetaKey l, MetaKey r) { return !(l == r); }
00065   friend constexpr bool operator!=(MetaKey l, MetaKey::Value r) { return !(l == r); }
00066   friend constexpr bool operator!=(MetaKey::Value l, MetaKey r) { return !(l == r); }
00067
00068   friend constexpr bool operator<(MetaKey l, MetaKey r) { return l.value < r.value; }
00069
00070   friend constexpr bool operator>=(MetaKey l, MetaKey r) { return !(l < r); }
00071   friend constexpr bool operator>=(MetaKey l, MetaKey::Value r) { return !(l < r); }
00072   friend constexpr bool operator>=(MetaKey::Value l, MetaKey r) { return !(l < r); }
00073
00074   friend constexpr bool operator>(MetaKey l, MetaKey r) { return r < l; }
00075   friend constexpr bool operator>(MetaKey l, MetaKey::Value r) { return r < l; }
00076   friend constexpr bool operator>(MetaKey::Value l, MetaKey r) { return r < l; }
00077
00078   friend constexpr bool operator<=(MetaKey l, MetaKey r) { return !(l > r); }
00079   friend constexpr bool operator<=(MetaKey l, MetaKey::Value r) { return !(l > r); }
00080   friend constexpr bool operator<=(MetaKey::Value l, MetaKey r) { return !(l > r); }
00081
00082   std::int32_t value;
00083 };
00084
00085 class Key
00086 {
00087 public:
00088   using value_type = std::int32_t;
00089
00090   enum Value : std::int32_t
00091   {
00092     NoKey = -1,
00093     // Now use « to for detecting special key + key press
00094
00095     // Begin ASCII (some ASCII names has been change to their Ctrl_+key part) the value is different
      to be able to do
00096     Ctrl_Arobase      = 0,
00097     Ctrl_A            = 1,
00098     Ctrl_B            = 2,
00099     Ctrl_C            = 3,
00100     Ctrl_D            = 4,
00101     Ctrl_E            = 5,
00102     Ctrl_F            = 6,
00103     Ctrl_G            = 7,
00104     Ctrl_H            = 8, /*corresponds to Backspace*/
00105     Ctrl_I            = 9, /*corresponds to Tab*/
00106     Ctrl_J            = 10,
00107     Ctrl_K            = 11,
00108     Ctrl_L            = 12,
00109     Ctrl_M            = 13, /*corresponds to Enter*/
00110     Ctrl_N            = 14,
00111     Ctrl_O            = 15,
00112     Ctrl_P            = 16,
00113     Ctrl_Q            = 17,
00114     Ctrl_R            = 18,
00115     Ctrl_S            = 19,
00116     Ctrl_T            = 20,
00117     Ctrl_U            = 21,
00118     Ctrl_V            = 22,
00119     Ctrl_W            = 23,
00120     Ctrl_X            = 24,
00121     Ctrl_Y            = 25,
00122     Ctrl_Z            = 26,
```

```
00123     Ctrl_OpenBracket  = 27, /*corresponds to Escape*/
00124     Ctrl_BackSlash    = 28,
00125     Ctrl_CloseBracket = 29,
00126     Ctrl_Caret        = 30,
00127     Ctrl_Underscore   = 31,
00128     Space             = 32,
00129     ExclamationMark   = 33,
00130     Quote             = 34,
00131     Hash              = 35,
00132     Dollar            = 36,
00133     Percent           = 37,
00134     Ampersand         = 38,
00135     Apostrophe        = 39,
00136     OpenParenthesis   = 40,
00137     CloseParenthesis  = 41,
00138     Asterisk          = 42,
00139     Plus              = 43,
00140     Comma             = 44,
00141     Hyphen            = 45,
00142     Minus             = 45,
00143     Period            = 46,
00144     Slash             = 47,
00145     Zero              = 48,
00146     One               = 49,
00147     Two               = 50,
00148     Three             = 51,
00149     Four              = 52,
00150     Five              = 53,
00151     Six               = 54,
00152     Seven             = 55,
00153     Eight             = 56,
00154     Nine              = 57,
00155     Colon             = 58,
00156     Semicolon         = 59,
00157     LessThan          = 60,
00158     OpenChevron       = 60,
00159     Equal             = 61,
00160     GreaterThan       = 62,
00161     CloseChevron      = 62,
00162     QuestionMark      = 63,
00163     Arobase           = 64,
00164     A                 = 65,
00165     B                 = 66,
00166     C                 = 67,
00167     D                 = 68,
00168     E                 = 69,
00169     F                 = 70,
00170     G                 = 71,
00171     H                 = 72,
00172     I                 = 73,
00173     J                 = 74,
00174     K                 = 75,
00175     L                 = 76,
00176     M                 = 77,
00177     N                 = 78,
00178     O                 = 79,
00179     P                 = 80,
00180     Q                 = 81,
00181     R                 = 82,
00182     S                 = 83,
00183     T                 = 84,
00184     U                 = 85,
00185     V                 = 86,
00186     W                 = 87,
00187     X                 = 88,
00188     Y                 = 89,
00189     Z                 = 90,
00190     OpenBracket       = 91,
00191     Backslash         = 92,
00192     CloseBracket      = 93,
00193     Caret             = 94,
00194     Underscore        = 95,
00195     GraveAccent       = 96,
00196     a                 = 97,
00197     b                 = 98,
00198     c                 = 99,
00199     d                 = 100,
00200     e                 = 101,
00201     f                 = 102,
00202     g                 = 103,
00203     h                 = 104,
00204     i                 = 105,
00205     j                 = 106,
00206     k                 = 107,
00207     l                 = 108,
00208     m                 = 109,
00209     n                 = 110,
```

```
00210     o                 = 111,
00211     p                 = 112,
00212     q                 = 113,
00213     r                 = 114,
00214     s                 = 115,
00215     t                 = 116,
00216     u                 = 117,
00217     v                 = 118,
00218     w                 = 119,
00219     x                 = 120,
00220     y                 = 121,
00221     z                 = 122,
00222     OpenBrace         = 123,
00223     VerticalBar       = 124,
00224     Close_Brace       = 125,
00225     Tilde             = 126,
00226     CTRL_QuestionMark = 127, /*corresponds to DEL*/
00227     // Very useful CTRL_* alternative names
00228     Null              = 0,
00229     Backspace         = 8,
00230     Tab               = 9,
00231     Enter             = 13,
00232     Esc               = 27,
00233     Del               = 127,
00234     //
00235     // End ASCII
00236     // Extended ASCII goes up to 255
00237     // Last Unicode codepage 0x10FFFF
00238     ArrowLeft         = 0x10FFFF + 1,
00239     ArrowRight        = 0x10FFFF + 2,
00240     ArrowUp           = 0x10FFFF + 3,
00241     ArrowDown         = 0x10FFFF + 4,
00242     Numeric5          = 0x10FFFF + 5,
00243     Home              = 0x10FFFF + 6,
00244     Insert            = 0x10FFFF + 7,
00245     End               = 0x10FFFF + 8,
00246     PageUp            = 0x10FFFF + 9,
00247     PageDown          = 0x10FFFF + 10,
00248     F1                = 0x10FFFF + 11,
00249     F2                = 0x10FFFF + 12,
00250     F3                = 0x10FFFF + 13,
00251     F4                = 0x10FFFF + 14,
00252     F5                = 0x10FFFF + 15,
00253     F6                = 0x10FFFF + 16,
00254     F7                = 0x10FFFF + 17,
00255     F8                = 0x10FFFF + 18,
00256     F9                = 0x10FFFF + 19,
00257     F10               = 0x10FFFF + 20,
00258     F11               = 0x10FFFF + 21,
00259     F12               = 0x10FFFF + 22,
00260     F13               = 0x10FFFF + 23,
00261     F14               = 0x10FFFF + 24,
00262     F15               = 0x10FFFF + 25,
00263     F16               = 0x10FFFF + 26,
00264     F17               = 0x10FFFF + 27,
00265     F18               = 0x10FFFF + 28,
00266     F19               = 0x10FFFF + 29,
00267     F20               = 0x10FFFF + 30,
00268     F21               = 0x10FFFF + 31,
00269     F22               = 0x10FFFF + 32,
00270     F23               = 0x10FFFF + 33,
00271     F24               = 0x10FFFF + 34,
00272     PrintScreen       = 0x10FFFF + 35,
00273     Menu              = 0x10FFFF + 36,
00274   };
00275
00276   constexpr Key() : value(NoKey) {}
00277   constexpr Key(const Key& key)         = default;
00278   inline Key& operator=(const Key& key) = default;
00279
00280   constexpr Key(const Value& v) : value(static_cast<std::int32_t>(v)) {}
00281   inline Key& operator=(const Value& v)
00282   {
00283     this->value = static_cast<std::int32_t>(v);
00284     return *this;
00285   }
00286
00287   explicit constexpr Key(char val) : value(static_cast<std::int32_t>(val)) {}
00288   inline Key& operator=(char val)
00289   {
00290     value = static_cast<std::int32_t>(val);
00291     return *this;
00292   }
00293
00294   constexpr Key(std::int32_t val) : value(val) {}
00295   inline Key& operator=(std::int32_t val)
00296   {
```

```
00297     value = val;
00298     return *this;
00299   }
00300
00301   explicit constexpr Key(std::size_t val) : value(static_cast<std::int32_t>(val)) {}
00302   inline Key& operator=(std::size_t val)
00303   {
00304     value = static_cast<std::int32_t>(val);
00305     return *this;
00306   }
00307
00308   explicit constexpr Key(char32_t val) : value(static_cast<std::int32_t>(val)) {}
00309   inline Key& operator=(char32_t val)
00310   {
00311     value = static_cast<std::int32_t>(val);
00312     return *this;
00313   }
00314
00315   constexpr operator std::int32_t() const { return this->value; }
00316
00317   friend constexpr bool operator==(Key l, Key r) { return l.value == r.value; }
00318   friend constexpr bool operator==(Key l, char r) { return l == Key(r); }
00319   friend constexpr bool operator==(char l, Key r) { return Key(l) == r; }
00320   friend constexpr bool operator==(Key l, char32_t r) { return l == Key(r); }
00321   friend constexpr bool operator==(char32_t l, Key r) { return Key(l) == r; }
00322   friend constexpr bool operator==(Key l, std::int32_t r) { return l == Key(r); }
00323   friend constexpr bool operator==(std::int32_t l, Key r) { return Key(l) == r; }
00324   friend constexpr bool operator==(Key l, std::size_t r) { return static_cast<std::size_t>(l.value) ==
      r; }
00325   friend constexpr bool operator==(std::size_t l, Key r) { return l ==
      static_cast<std::size_t>(r.value); }
00326
00327   friend constexpr bool operator!=(Key l, Key r) { return !(l == r); }
00328   friend constexpr bool operator!=(Key l, char r) { return !(l == r); }
00329   friend constexpr bool operator!=(char l, Key r) { return !(l == r); }
00330   friend constexpr bool operator!=(Key l, char32_t r) { return !(l == r); }
00331   friend constexpr bool operator!=(char32_t l, Key r) { return !(l == r); }
00332   friend constexpr bool operator!=(Key l, std::int32_t r) { return !(l == r); }
00333   friend constexpr bool operator!=(std::int32_t l, Key r) { return !(l == r); }
00334   friend constexpr bool operator!=(Key l, std::size_t r) { return !(l == r); }
00335   friend constexpr bool operator!=(std::size_t l, Key r) { return !(l == r); }
00336
00337   friend constexpr bool operator<(Key l, Key r) { return l.value < r.value; }
00338   friend constexpr bool operator<(Key l, char r) { return l < Key(r); }
00339   friend constexpr bool operator<(char l, Key r) { return Key(l) < r; }
00340   friend constexpr bool operator<(Key l, char32_t r) { return l < Key(r); }
00341   friend constexpr bool operator<(char32_t l, Key r) { return Key(l) < r; }
00342   friend constexpr bool operator<(Key l, std::int32_t r) { return l < Key(r); }
00343   friend constexpr bool operator<(std::int32_t l, Key r) { return Key(l) < r; }
00344   friend constexpr bool operator<(Key l, std::size_t r) { return static_cast<std::size_t>(l.value) <
      r; }
00345   friend constexpr bool operator<(std::size_t l, Key r) { return
      l < static_cast<std::size_t>(r.value); }
00346
00347   friend constexpr bool operator>=(Key l, Key r) { return !(l < r); }
00348   friend constexpr bool operator>=(Key l, char r) { return !(l < r); }
00349   friend constexpr bool operator>=(char l, Key r) { return !(l < r); }
00350   friend constexpr bool operator>=(Key l, char32_t r) { return !(l < r); }
00351   friend constexpr bool operator>=(char32_t l, Key r) { return !(l < r); }
00352   friend constexpr bool operator>=(Key l, std::int32_t r) { return !(l < r); }
00353   friend constexpr bool operator>=(std::int32_t l, Key r) { return !(l < r); }
00354   friend constexpr bool operator>=(Key l, std::size_t r) { return !(l < r); }
00355   friend constexpr bool operator>=(std::size_t l, Key r) { return !(l < r); }
00356
00357   friend constexpr bool operator>(Key l, Key r) { return r < l; }
00358   friend constexpr bool operator>(Key l, char r) { return r < l; }
00359   friend constexpr bool operator>(char l, Key r) { return r < l; }
00360   friend constexpr bool operator>(Key l, char32_t r) { return r < l; }
00361   friend constexpr bool operator>(char32_t l, Key r) { return r < l; }
00362   friend constexpr bool operator>(Key l, std::int32_t r) { return r < l; }
00363   friend constexpr bool operator>(std::int32_t l, Key r) { return r < l; }
00364   friend constexpr bool operator>(Key l, std::size_t r) { return r < l; }
00365   friend constexpr bool operator>(std::size_t l, Key r) { return r < l; }
00366
00367   friend constexpr bool operator<=(Key l, Key r) { return !(l > r); }
00368   friend constexpr bool operator<=(Key l, char r) { return !(l > r); }
00369   friend constexpr bool operator<=(char l, Key r) { return !(l > r); }
00370   friend constexpr bool operator<=(Key l, char32_t r) { return !(l > r); }
00371   friend constexpr bool operator<=(char32_t l, Key r) { return !(l > r); }
00372   friend constexpr bool operator<=(Key l, std::int32_t r) { return !(l > r); }
00373   friend constexpr bool operator<=(std::int32_t l, Key r) { return !(l > r); }
00374   friend constexpr bool operator<=(Key l, std::size_t r) { return !(l > r); }
00375   friend constexpr bool operator<=(std::size_t l, Key r) { return !(l > r); }
00376
00377   constexpr bool iscntrl() const { return (*this >= Key::Null && *this <= Key::Ctrl_Underscore) ||
      *this == Key::Del; }
00378   constexpr bool isblank() const { return *this == Key::Tab || *this == Key::Space; }
```

```
00379   constexpr bool isspace() const { return this->isblank() || (*this >= Key::Ctrl_J && *this <=
      Key::Enter); }
00380   constexpr bool isupper() const { return *this >= Key::A && *this <= Key::Z; }
00381   constexpr bool islower() const { return *this >= Key::a && *this <= Key::z; }
00382   constexpr bool isalpha() const { return (this->isupper() || this->islower()); }
00383   constexpr bool isdigit() const { return *this >= Key::Zero && *this <= Key::Nine; }
00384   constexpr bool isxdigit() const { return this->isdigit() || (*this >= Key::A && *this <= Key::F) ||
      (*this >= Key::a && *this <= Key::f); }
00385   constexpr bool isalnum() const { return (this->isdigit() || this->isalpha()); }
00386   constexpr bool ispunct() const { return (*this >= Key::ExclamationMark && *this <= Key::Slash) ||
      (*this >= Key::Colon && *this <= Key::Arobase) || (*this >= Key::OpenBracket && *this <=
      Key::GraveAccent) || (*this >= Key::OpenBrace && *this <= Key::Tilde); }
00387   constexpr bool isgraph() const { return (this->isalnum() || this->ispunct()); }
00388   constexpr bool isprint() const { return (this->isgraph() || *this == Key::Space); }
00389   constexpr bool isunicode() const { return *this >= Key::Null && this->value <= 0x10FFFFL; }
00390   constexpr Key  tolower() const { return (this->isalpha() && this->isupper()) ? Key(this->value + 32)
      : *this; }
00391   constexpr Key  toupper() const { return (this->isalpha() && this->islower()) ? Key(this->value - 32)
      : *this; }
00392
00393   // Detect if *this is convertible to ANSII
00394   constexpr bool isASCII() const { return *this >= Key::Null && *this <= Key::Del; }
00395
00396   // Detect if *this is convertible to Extended ANSII
00397   constexpr bool isExtendedASCII() const { return *this >= Key::Null && this->value <= 255L; }
00398
00399   // Detect if *this has CTRL+*
00400   constexpr bool hasCtrlAll() const { return this->iscntrl() || ((this->value &
      static_cast<std::int32_t>(MetaKey::Value::Ctrl)) == static_cast<std::int32_t>(MetaKey::Value::Ctrl));
      }
00401
00402   // Detect if *this has CTRL+* (excluding the CTRL+* that can be access with standard *this Tab
      Backspace Enter...)
00403   constexpr bool hasCtrl() const
00404   {
00405     // Need to suppress the TAB etc...
00406     return ((this->iscntrl() || this->hasCtrlAll()) && *this != Key::Backspace && *this != Key::Tab &&
      *this != Key::Esc && *this != Key::Enter && *this != Key::Del);
00407   }
00408
00409   // Detect if key has ALT+*
00410   constexpr bool hasAlt() const { return (this->value &
      static_cast<std::int32_t>(MetaKey::Value::Alt)) == static_cast<std::int32_t>(MetaKey::Value::Alt); }
00411
00412   constexpr bool empty() const { return (this->value == Key::NoKey); }
00413
00414   void        append_name(std::string& strOut) const;
00415   std::string name() const;
00416   std::string str() const;
00417
00418   // member variable value
00419   // cannot be Key::Value and has to be std::int32_t because it can also have numbers
00420   // that are not named within the enum. Otherwise it would be undefined behaviour
00421   std::int32_t value;
00422 };
00423
00424 constexpr bool operator==(Key l, MetaKey r) { return static_cast<std::int32_t>(l) ==
      static_cast<std::int32_t>(r); }
00425 constexpr bool operator==(MetaKey l, Key r) { return static_cast<std::int32_t>(l) ==
      static_cast<std::int32_t>(r); }
00426
00427 constexpr bool operator<(MetaKey l, Key r) { return static_cast<std::int32_t>(l) <
      static_cast<std::int32_t>(r); }
00428 constexpr bool operator<(Key l, MetaKey r) { return static_cast<std::int32_t>(l) <
      static_cast<std::int32_t>(r); }
00429
00430 constexpr bool operator!=(Key l, MetaKey r) { return static_cast<std::int32_t>(l) !=
      static_cast<std::int32_t>(r); }
00431 constexpr bool operator!=(MetaKey l, Key r) { return static_cast<std::int32_t>(l) !=
      static_cast<std::int32_t>(r); }
00432
00433 constexpr bool operator>=(MetaKey l, Key r) { return static_cast<std::int32_t>(l) >=
      static_cast<std::int32_t>(r); }
00434 constexpr bool operator>=(Key l, MetaKey r) { return static_cast<std::int32_t>(l) >=
      static_cast<std::int32_t>(r); }
00435
00436 constexpr bool operator>(MetaKey l, Key r) { return static_cast<std::int32_t>(l) >
      static_cast<std::int32_t>(r); }
00437 constexpr bool operator>(Key l, MetaKey r) { return static_cast<std::int32_t>(l) >
      static_cast<std::int32_t>(r); }
00438
00439 constexpr bool operator<=(MetaKey l, Key r) { return static_cast<std::int32_t>(l) <=
      static_cast<std::int32_t>(r); }
00440 constexpr bool operator<=(Key l, MetaKey r) { return static_cast<std::int32_t>(l) <=
      static_cast<std::int32_t>(r); }
00441
00442 constexpr Key operator+(MetaKey metakey, Key key) { return Key(key.value + ((metakey ==
```

```
      MetaKey::Value::Ctrl && !key.hasCtrlAll() && !key.empty()) ?
      static_cast<std::int32_t>(MetaKey::Value::Ctrl) : 0) + ((metakey == MetaKey::Value::Alt &&
      !key.hasAlt() && !key.empty()) ? static_cast<std::int32_t>(MetaKey::Value::Alt) : 0)); }
00443 constexpr Key operator+(Key key, MetaKey meta) { return meta + key; }
00444
00445 constexpr Key operator+(MetaKey::Value l, Key r) { return MetaKey(l) + r; }
00446 constexpr Key operator+(Key l, MetaKey::Value r) { return l + MetaKey(r); }
00447 constexpr Key operator+(MetaKey::Value l, Key::value_type r) { return MetaKey(l) + Key(r); }
00448 constexpr Key operator+(Key::value_type l, MetaKey::Value r) { return Key(l) + MetaKey(r); }
00449
00450 }  // namespace Term
```

9.41 cpp-terminal/mouse.cpp File Reference

```
#include "cpp-terminal/mouse.hpp"
```

9.42 mouse.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/mouse.hpp"
00011
00012 Term::Button::Action Term::Button::action() const noexcept { return m_action; }
00013
00014 Term::Button::Type Term::Button::type() const noexcept { return m_type; }
00015
00016 bool Term::Button::operator==(const Term::Button& button) const { return (m_action == button.m_action)
      && (m_type == button.m_type); }
00017
00018 bool Term::Button::operator!=(const Term::Button& button) const { return !(*this == button); }
00019
00020 bool Term::Mouse::is(const Term::Button::Type& type, const Term::Button::Action& action) const
      noexcept { return (m_buttons.type() == type) && (m_buttons.action() == action); }
00021
00022 bool Term::Mouse::is(const Term::Button::Type& type) const noexcept { return m_buttons.type() == type;
      }
00023
00024 Term::Button Term::Mouse::getButton() const noexcept { return m_buttons; }
00025
00026 std::size_t Term::Mouse::row() const noexcept { return static_cast<std::size_t>(m_row); }
00027
00028 std::size_t Term::Mouse::column() const noexcept { return static_cast<std::size_t>(m_column); }
00029
00030 bool Term::Mouse::operator==(const Term::Mouse& mouse) const { return (m_row == mouse.m_row) &&
      (m_column == mouse.m_column) && (m_buttons == mouse.m_buttons); }
00031 bool Term::Mouse::operator!=(const Term::Mouse& mouse) const { return !(*this == mouse); }
```

9.43 cpp-terminal/mouse.hpp File Reference

```
#include <cstddef>
#include <cstdint>
```

Classes

- class Term::Button
- class Term::Mouse

Namespaces

- namespace Term

9.44 mouse.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstddef>
00013 #include <cstdint>
00014
00015 namespace Term
00016 {
00017
00018 class Button
00019 {
00020 public:
00021   enum class Type : std::int8_t
00022   {
00023     None = -1,
00024     Left,
00025     Wheel,
00026     Right,
00027     Button1,
00028     Button2,
00029     Button3,
00030     Button4,
00031     Button5,
00032     Button6,
00033     Button7,
00034     Button8,
00035   };
00036   enum class Action : std::int8_t
00037   {
00038     None = 0,
00039     Pressed,
00040     Released,
00041     DoubleClicked,
00042     RolledUp,
00043     RolledDown,
00044     ToRight,
00045     ToLeft,
00046   };
00047   Button() = default;
00048   Button(const Term::Button::Type& type, const Term::Button::Action& action) : m_type(type),
      m_action(action) {}
00049   Term::Button::Action action() const noexcept;
00050   Term::Button::Type   type() const noexcept;
00051   bool                 operator==(const Term::Button& button) const;
00052   bool                 operator!=(const Term::Button& button) const;
00053
00054 private:
00055   Term::Button::Type   m_type{Term::Button::Type::None};
00056   Term::Button::Action m_action{Term::Button::Action::None};
00057 };
00058
00059 class Mouse
00060 {
00061 public:
00062   Mouse() = default;
00063   Mouse(const Term::Button& buttons, const std::uint16_t& row, const std::uint16_t& column) :
      m_buttons(buttons), m_row(row), m_column(column) {}
00064   std::size_t  row() const noexcept;
00065   std::size_t  column() const noexcept;
00066   Term::Button getButton() const noexcept;
00067   bool         is(const Term::Button::Type& type, const Term::Button::Action& action) const noexcept;
00068   bool         is(const Term::Button::Type& type) const noexcept;
00069   bool         operator==(const Term::Mouse& mouse) const;
00070   bool         operator!=(const Term::Mouse& mouse) const;
00071
00072 private:
```

```
00073   Term::Button  m_buttons;
00074   std::uint16_t m_row{0};
00075   std::uint16_t m_column{0};
00076 };
00077
00078 }  // namespace Term
```

9.45 cpp-terminal/options.cpp File Reference

```
#include "cpp-terminal/options.hpp"
#include <algorithm>
```

9.46 options.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/options.hpp"
00011
00012 #include <algorithm>
00013
00014 Term::Options::Options(const std::initializer_list<Term::Option>& option) : m_Options(option) {
      clean(); }
00015
00016 bool Term::Options::operator==(const Options& options) { return m_Options == options.m_Options; }
00017 bool Term::Options::operator!=(const Options& options) { return !(m_Options == options.m_Options); }
00018
00019 void Term::Options::clean()
00020 {
00021   std::vector<Term::Option> cleaned;
00022   std::sort(m_Options.begin(), m_Options.end());
00023   while(!m_Options.empty())
00024   {
00025     const std::size_t count      = std::count(m_Options.begin(), m_Options.end(), m_Options[0]);
00026     const std::size_t anti_count = std::count(m_Options.begin(), m_Options.end(),
      static_cast<Term::Option>(-1 * static_cast<std::int16_t>(m_Options[0])));
00027     if(count > anti_count) { cleaned.emplace_back(m_Options[0]); }
00028     else if(count < anti_count) { cleaned.emplace_back(static_cast<Term::Option>(-1 *
      static_cast<std::int16_t>(m_Options[0]))); }
00029     m_Options.erase(std::remove(m_Options.begin(), m_Options.end(), static_cast<Term::Option>(-1 *
      static_cast<std::int16_t>(m_Options[0]))), m_Options.end());
00030     m_Options.erase(std::remove(m_Options.begin(), m_Options.end(), m_Options[0]), m_Options.end());
00031   }
00032   m_Options = cleaned;
00033 }
00034
00035 bool Term::Options::has(const Option& option) const noexcept { return std::find(m_Options.begin(),
      m_Options.end(), option) != m_Options.end(); }
```

9.47 cpp-terminal/options.hpp File Reference

```
#include <cstdint>
#include <initializer_list>
#include <vector>
```

Classes

- class Term::Options

Namespaces

- namespace Term

Enumerations

- enum class Term::Option : std::int16_t {
  Term::Raw = 1 , Term::Cooked = -1 , Term::ClearScreen = 2 , Term::NoClearScreen = -2 ,
  Term::SignalKeys = 3 , Term::NoSignalKeys = -3 , Term::Cursor = 4 , Term::NoCursor = -4 }

  Option to set-up the terminal.

9.48 options.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstdint>
00013 #include <initializer_list>
00014 #include <vector>
00015
00016 namespace Term
00017 {
00018
00022 enum class Option : std::int16_t
00023 {
00024   Raw           = 1,
00025   Cooked        = -1,
00026   ClearScreen   = 2,
00027   NoClearScreen = -2,
00028   SignalKeys    = 3,
00029   NoSignalKeys  = -3,
00030   Cursor        = 4,
00031   NoCursor      = -4
00032 };
00033
00034 class Options
00035 {
00036 public:
00037   Options() = default;
00038   Options(const std::initializer_list<Term::Option>& option);
00039   template<typename... Args> explicit Options(const Args&&... args) :
      m_Options(std::initializer_list<Term::Option>{args...}) { clean(); }
00040
00041   bool operator==(const Options& options);
00042   bool operator!=(const Options& options);
00043   bool has(const Option& option) const noexcept;
00044
00045 private:
00046   void                       clean();
00047   std::vector<Term::Option> m_Options;
00048 };
00049
00050 }  // namespace Term
```

9.49 cpp-terminal/args.cpp File Reference

```
#include "cpp-terminal/args.hpp"
```

Namespaces

- namespace Term

9.50 args.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/args.hpp"
00011
00012 namespace Term
00013 {
00014
00015 Term::Arguments::Arguments() {}
00016
00017 Term::Argc::Argc() {}
00018
00019 Term::Argc::operator unsigned int() { return static_cast<unsigned int>(Term::Arguments::argc()); }
00020
00021 Term::Argc::operator unsigned int() const { return static_cast<unsigned int>(Term::Arguments::argc());
      }
00022
00023 std::string Term::Arguments::operator[](const std::size_t& arg) const { return m_args[arg]; }
00024
00025 }  // namespace Term
```

9.51 cpp-terminal/private/args.cpp File Reference

```
#include "cpp-terminal/args.hpp"
#include <ios>
#include <memory>
#include <windows.h>
#include <processenv.h>
#include "cpp-terminal/private/unicode.hpp"
```

9.52 args.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/args.hpp"
00011
00012 #include <ios>
00013
00014 #if defined(_WIN32)
00015   #include <memory>
00016 // clang-format off
00017   #include <windows.h>
00018   #include <processenv.h>
00019   #include "cpp-terminal/private/unicode.hpp"
00020 // clang-format on
00021 #elif defined(__APPLE__)
00022   #include <crt_externs.h>
```

```
00023 #else
00024   #include <algorithm>
00025   #include <cstddef>
00026   #include <fstream>
00027   #include <limits>
00028 #endif
00029
00030 void Term::Arguments::parse()
00031 {
00032   if(m_parsed) { return; }
00033 #if defined(_WIN32)
00034   int                                          argc{0};
00035   std::unique_ptr<LPWSTR[], void (*)(wchar_t**)> wargv = std::unique_ptr<LPWSTR[], void
      (*)(wchar_t**)>(CommandLineToArgvW(GetCommandLineW(), &argc), [](wchar_t** ptr) { LocalFree(ptr); });
00036   if(wargv == nullptr)
00037   {
00038     m_parsed = false;
00039     return;
00040   }
00041   else
00042   {
00043     m_args.reserve(static_cast<std::size_t>(argc));
00044     for(std::size_t i = 0; i != static_cast<std::size_t>(argc); ++i) {
      m_args.push_back(Term::Private::to_narrow(&wargv.get()[i][0])); }
00045     m_parsed = true;
00046   }
00047 #elif defined(__APPLE__)
00048   std::size_t argc{static_cast<std::size_t>(*_NSGetArgc())};
00049   m_args.reserve(argc);
00050   char** argv{*_NSGetArgv()};
00051   for(std::size_t i = 0; i != argc; ++i) { m_args.push_back(argv[i]); }
00052   m_parsed = true;
00053 #else
00054   std::string                cmdline;
00055   std::fstream               file_stream;
00056   const std::fstream::iostate old_iostate{file_stream.exceptions()};
00057   try
00058   {
00059     file_stream.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00060     file_stream.open("/proc/self/cmdline", std::fstream::in | std::fstream::binary);
00061     file_stream.ignore(std::numeric_limits<std::streamsize>::max());
00062     cmdline.resize(static_cast<std::size_t>(file_stream.gcount()));
00063     file_stream.seekg(0, std::ios_base::beg);
00064     file_stream.get(&cmdline[0], static_cast<std::streamsize>(cmdline.size()));
      //NOLINT(readability-container-data-pointer)
00065     file_stream.exceptions(old_iostate);
00066     if(file_stream.is_open()) { file_stream.close(); }
00067     m_args.reserve(static_cast<std::size_t>(std::count(cmdline.begin(), cmdline.end(), '\0')));
00068     for(std::string::iterator it = cmdline.begin(); it != cmdline.end(); it = std::find(it,
      cmdline.end(), '\0') + 1) { m_args.emplace_back(cmdline.data() + (it - cmdline.begin())); }
00069     m_parsed = true;
00070   }
00071   catch(...)
00072   {
00073     file_stream.exceptions(old_iostate);
00074     if(file_stream.is_open()) { file_stream.close(); }
00075     m_args.clear();
00076     m_parsed = false;
00077   }
00078 #endif
00079 }
00080
00081 std::size_t Term::Arguments::argc()
00082 {
00083   parse();
00084   return m_args.size();
00085 }
00086
00087 std::vector<std::string> Term::Arguments::argv()
00088 {
00089   parse();
00090   return m_args;
00091 }
00092
00093 bool Term::Arguments::m_parsed = false;
00094
00095 std::vector<std::string> Term::Arguments::m_args = std::vector<std::string>();
      //NOLINT(fuchsia-statically-constructed-objects)
```

9.53 cpp-terminal/private/blocking_queue.cpp File Reference

```
#include "cpp-terminal/private/blocking_queue.hpp"
```

9.54 blocking_queue.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/blocking_queue.hpp"
00011
00012 Term::Event Term::Private::BlockingQueue::pop()
00013 {
00014   const std::lock_guard<std::mutex> lock(m_mutex);
00015   Term::Event                       value = this->m_queue.front();
00016   m_queue.pop();
00017   return value;
00018 }
00019
00020 void Term::Private::BlockingQueue::push(const Term::Event& value, const std::size_t& occurrence)
00021 {
00022   for(std::size_t i = 0; i != occurrence; ++i)
00023   {
00024     const std::lock_guard<std::mutex> lock(m_mutex);
00025     m_queue.push(value);
00026     m_cv.notify_all();
00027   }
00028 }
00029
00030 void Term::Private::BlockingQueue::push(const Term::Event&& value, const std::size_t& occurrence)
00031 {
00032   for(std::size_t i = 0; i != occurrence; ++i)
00033   {
00034     const std::lock_guard<std::mutex> lock(m_mutex);
00035     m_queue.push(value);
00036     m_cv.notify_all();
00037   }
00038 }
00039
00040 bool Term::Private::BlockingQueue::empty()
00041 {
00042   const std::lock_guard<std::mutex> lock(m_mutex);
00043   return m_queue.empty();
00044 }
00045
00046 std::size_t Term::Private::BlockingQueue::size()
00047 {
00048   const std::lock_guard<std::mutex> lock(m_mutex);
00049   return m_queue.size();
00050 }
00051
00052 void Term::Private::BlockingQueue::wait_for_events(std::unique_lock<std::mutex>& lock) {
      m_cv.wait(lock); }
```

9.55 cpp-terminal/private/blocking_queue.hpp File Reference

```
#include "cpp-terminal/event.hpp"
#include <condition_variable>
#include <mutex>
#include <queue>
```

Classes

- class Term::Private::BlockingQueue

Namespaces

- namespace Term
- namespace Term::Private

9.56 blocking_queue.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/event.hpp"
00013
00014 #include <condition_variable>
00015 #include <mutex>
00016 #include <queue>
00017
00018 namespace Term
00019 {
00020
00021 namespace Private
00022 {
00023
00024 class BlockingQueue
00025 {
00026 public:
00027   ~BlockingQueue()                                    = default;
00028   BlockingQueue()                                     = default;
00029   BlockingQueue(const BlockingQueue& other)           = delete;
00030   BlockingQueue(BlockingQueue&& other)                = delete;
00031   BlockingQueue& operator=(const BlockingQueue& other) = delete;
00032   BlockingQueue& operator=(BlockingQueue&& other)      = delete;
00033   Term::Event    pop();
00034   void           push(const Term::Event& value, const std::size_t& occurrence = 1);
00035   void           push(const Term::Event&& value, const std::size_t& occurrence = 1);
00036   bool           empty();
00037   std::size_t    size();
00038   void           wait_for_events(std::unique_lock<std::mutex>& lock);
00039
00040 private:
00041   std::mutex              m_mutex;
00042   std::queue<Term::Event> m_queue;
00043   std::condition_variable m_cv;
00044 };
00045
00046 }  // namespace Private
00047 }  // namespace Term
```

9.57 cpp-terminal/private/conversion.cpp File Reference

```
#include "cpp-terminal/private/conversion.hpp"
#include "cpp-terminal/exception.hpp"
#include <array>
#include <string>
```

Namespaces

- namespace Term
- namespace Term::Private

Functions

- std::uint8_t Term::Private::utf8_decode_step (std::uint8_t state, std::uint8_t octet, std::uint32_t *cpp)
- std::u32string Term::Private::utf8_to_utf32 (const std::string &str)
- bool Term::Private::is_valid_utf8_code_unit (const std::string &str)

9.58 conversion.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/conversion.hpp"
00011
00012 #include "cpp-terminal/exception.hpp"
00013
00014 #include <array>
00015 #include <string>
00016
00017 namespace Term
00018 {
00019 namespace Private
00020 {
00021
00022 static constexpr std::uint8_t UTF8_ACCEPT{0};
00023 static constexpr std::uint8_t UTF8_REJECT{0xf};
00024
00025 std::uint8_t utf8_decode_step(std::uint8_t state, std::uint8_t octet, std::uint32_t* cpp)
00026 {
00027   static const constexpr std::array<std::uint32_t, 0x10> utf8ClassTab{0x88888888UL, 0x88888888UL,
      0x99999999UL, 0x99999999UL, 0xaaaaaaaaUL, 0xaaaaaaaaUL, 0xaaaaaaaaUL, 0xaaaaaaaaUL, 0x222222ffUL,
      0x22222222UL, 0x22222222UL, 0x22222222UL, 0x3333333bUL, 0x33433333UL, 0xfff5666cUL, 0xffffffffUL};
00028
00029   static const constexpr std::array<std::uint32_t, 0x10> utf8StateTab{0xfffffff0UL, 0xffffffffUL,
      0xfffffff1UL, 0xfffffff3UL, 0xfffffff4UL, 0xfffffff7UL, 0xfffffff6UL, 0xffffffffUL, 0x33f11f0fUL,
      0xf3311f0fUL, 0xf33f110fUL, 0xfffffff2UL, 0xfffffff5UL, 0xffffffffUL, 0xffffffffUL, 0xffffffffUL};
00030
00031   const std::uint8_t reject{static_cast<std::uint8_t>(state » 3UL)};
00032   const std::uint8_t nonAscii{static_cast<std::uint8_t>(octet » 7UL)};
00033   const std::uint8_t class_{static_cast<std::uint8_t>(!nonAscii ? 0 : (0xf & (utf8ClassTab[(octet » 3)
      & 0xf] » (4 * (octet & 7)))))};
00034
00035   *cpp = (state == UTF8_ACCEPT ? (octet & (0xffU » class_)) : ((octet & 0x3fU) | (*cpp « 6)));
00036
00037   return (reject ? 0xf : (0xf & (utf8StateTab[class_] » (4 * (state & 7)))));
00038 }
00039
00040 std::u32string utf8_to_utf32(const std::string& str)
00041 {
00042   std::uint32_t  codepoint{0};
00043   std::uint8_t   state{UTF8_ACCEPT};
00044   std::u32string ret;
00045   for(char idx: str)
00046   {
00047     state = utf8_decode_step(state, static_cast<std::uint8_t>(idx), &codepoint);
00048     if(state == UTF8_ACCEPT) { ret.push_back(codepoint); }
00049     else if(state == UTF8_REJECT) { throw Term::Exception("Invalid byte in UTF8 encoded string"); }
00050   }
00051   if(state != UTF8_ACCEPT) { throw Term::Exception("Expected more bytes in UTF8 encoded string"); }
00052   return ret;
00053 }
00054
00055 bool is_valid_utf8_code_unit(const std::string& str)
00056 {
00057   static const constexpr std::uint8_t b10000000{128};
00058   static const constexpr std::uint8_t b11000000{192};
00059   static const constexpr std::uint8_t b11100000{224};
00060   static const constexpr std::uint8_t b11110000{240};
00061   static const constexpr std::uint8_t b11111000{248};
00062   switch(str.size())
00063   {
00064     case 1: return (static_cast<std::uint8_t>(str[0]) & b10000000) == 0;
00065     case 2: return ((static_cast<std::uint8_t>(str[0]) & b11100000) == b11000000) &&
      ((static_cast<std::uint8_t>(str[1]) & b11000000) == b10000000);
00066     case 3: return ((static_cast<std::uint8_t>(str[0]) & b11110000) == b11100000) &&
      ((static_cast<std::uint8_t>(str[1]) & b11000000) == b10000000) && ((static_cast<std::uint8_t>(str[2])
      & b11000000) == b10000000);
00067     case 4: return ((static_cast<std::uint8_t>(str[0]) & b11111000) == b11110000) &&
      ((static_cast<std::uint8_t>(str[1]) & b11000000) == b10000000) && ((static_cast<std::uint8_t>(str[2])
      & b11000000) == b10000000) && ((static_cast<std::uint8_t>(str[3]) & b11000000) == b10000000);
00068     default: return false;
00069   }
00070 }
00071
00072 }  // namespace Private
00073
```

```
00074 }  // namespace Term
```

9.59 cpp-terminal/private/conversion.hpp File Reference

```
#include <cstdint>
#include <string>
#include <vector>
```

Namespaces

- namespace Term
- namespace Term::Private

Functions

- std::uint8_t Term::Private::utf8_decode_step (std::uint8_t state, std::uint8_t octet, std::uint32_t *cpp)
- std::u32string Term::Private::utf8_to_utf32 (const std::string &str)
- bool Term::Private::is_valid_utf8_code_unit (const std::string &str)

9.60 conversion.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstdint>
00013 #include <string>
00014 #include <vector>
00015
00016 namespace Term
00017 {
00018 namespace Private
00019 {
00020
00021 std::uint8_t utf8_decode_step(std::uint8_t state, std::uint8_t octet, std::uint32_t* cpp);
00022
00023 std::u32string utf8_to_utf32(const std::string& str);
00024
00025 bool is_valid_utf8_code_unit(const std::string& str);
00026
00027 }  // namespace Private
00028
00029 }  // namespace Term
```

9.61 cpp-terminal/cursor.cpp File Reference

```
#include "cpp-terminal/cursor.hpp"
```

9.62 cursor.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/cursor.hpp"
00011
00012 Term::Cursor::Cursor(const std::size_t& row, const std::size_t& column) : m_position({row, column}) {}
00013
00014 std::size_t Term::Cursor::row() const { return m_position.first; }
00015
00016 std::size_t Term::Cursor::column() const { return m_position.second; }
00017
00018 bool Term::Cursor::empty() const { return (0 == m_position.first) && (0 == m_position.second); }
00019
00020 void Term::Cursor::setRow(const std::size_t& row) { m_position.first = row; }
00021
00022 void Term::Cursor::setColum(const std::size_t& column) { m_position.second = column; }
00023
00024 bool Term::Cursor::operator==(const Term::Cursor& cursor) const { return (this->row() == cursor.row())
      && (this->column() == cursor.column()); }
00025
00026 bool Term::Cursor::operator!=(const Term::Cursor& cursor) const { return !(*this == cursor); }
00027
00028 std::string Term::cursor_off() { return "\u001b[?25l"; }
00029
00030 std::string Term::cursor_on() { return "\u001b[?25h"; }
00031
00032 std::string Term::cursor_move(const std::size_t& row, const std::size_t& column) { return "\u001b[" +
      std::to_string(row) + ';' + std::to_string(column) + 'H'; }
00033
00034 std::string Term::cursor_up(const std::size_t& rows) { return "\u001b[" + std::to_string(rows) + 'A';
      }
00035
00036 std::string Term::cursor_down(const std::size_t& rows) { return "\u001b[" + std::to_string(rows) +
      'B'; }
00037
00038 std::string Term::cursor_right(const std::size_t& columns) { return "\u001b[" +
      std::to_string(columns) + 'C'; }
00039
00040 std::string Term::cursor_left(const std::size_t& columns) { return "\u001b[" + std::to_string(columns)
      + 'D'; }
00041
00042 std::string Term::cursor_position_report() { return "\u001b[6n"; }
00043
00044 std::string Term::clear_eol() { return "\u001b[K"; }
```

9.63 cpp-terminal/private/cursor.cpp File Reference

```
#include "cpp-terminal/cursor.hpp"
#include "file_initializer.hpp"
#include <windows.h>
#include "cpp-terminal/private/file.hpp"
```

9.64 cursor.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
```

```
00010 #include "cpp-terminal/cursor.hpp"
00011
00012 #include "file_initializer.hpp"
00013
00014 #if defined(_WIN32)
00015   #include <windows.h>
00016 #else
00017   #include <sys/ioctl.h>
00018   #include <termios.h>
00019 #endif
00020
00021 #include "cpp-terminal/private/file.hpp"
00022
00023 Term::Cursor Term::cursor_position()
00024 {
00025   static const Term::Private::FileInitializer files_init;
00026   if(Term::Private::in.null()) { return {}; }
00027 #if defined(_WIN32)
00028   CONSOLE_SCREEN_BUFFER_INFO inf;
00029   if(GetConsoleScreenBufferInfo(Private::out.handle(), &inf)) return
      Term::Cursor(static_cast<std::size_t>(inf.dwCursorPosition.Y + 1),
      static_cast<std::size_t>(inf.dwCursorPosition.X + 1));
00030   else
00031     return Term::Cursor(0, 0);
00032 #else
00033   std::string ret;
00034   std::size_t nread{0};
00035   Term::Private::in.lockIO();
00036   // Hack to be sure we can do this all the time "Cooked" or "Raw" mode
00037   ::termios actual;
00038   if(!Private::out.null())
00039   {
00040     if(tcgetattr(Private::out.fd(), &actual) == -1) { return {}; }
00041   }
00042   ::termios raw = actual;
00043   // Put terminal in raw mode
00044   raw.c_iflag &= ~(BRKINT | ICRNL | INPCK | ISTRIP | IXON);
00045   // This disables output post-processing, requiring explicit \r\n. We
00046   // keep it enabled, so that in C++, one can still just use std::endl
00047   // for EOL instead of "\r\n".
00048   // raw.c_oflag &= ~(OPOST);
00049   raw.c_lflag &= ~(ECHO | ICANON | IEXTEN);
00050   raw.c_lflag &= ~ISIG;
00051   raw.c_cc[VMIN]  = 1;
00052   raw.c_cc[VTIME] = 0;
00053   if(!Private::out.null()) tcsetattr(Private::out.fd(), TCSAFLUSH, &raw);
00054   Term::Private::out.write(Term::cursor_position_report());
00055   while(nread == 0) { ::ioctl(Private::in.fd(), FIONREAD, &nread); }
00056   ret = Term::Private::in.read();
00057   tcsetattr(Private::out.fd(), TCSAFLUSH, &actual);
00058   Term::Private::in.unlockIO();
00059   try
00060   {
00061     if(ret[0] == '\033' && ret[1] == '[' && ret[ret.size() - 1] == 'R')
00062     {
00063       std::size_t found = ret.find(';', 2);
00064       if(found != std::string::npos) { return Cursor(std::stoi(ret.substr(2, found - 2)),
      std::stoi(ret.substr(found + 1, ret.size() - (found + 2)))); }
00065       return {};
00066     }
00067     return {};
00068   }
00069   catch(...)
00070   {
00071     return {};
00072   }
00073 #endif
00074 }
```

9.65 cpp-terminal/private/env.cpp File Reference

```
#include "cpp-terminal/private/env.hpp"
#include "cpp-terminal/private/unicode.hpp"
```

9.66 env.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/env.hpp"
00011
00012 #include "cpp-terminal/private/unicode.hpp"
00013
00014 std::pair<bool, std::string> Term::Private::getenv(const std::string& key)
00015 {
00016 #if defined(_WIN32)
00017   std::size_t size{0};
00018   _wgetenv_s(&size, nullptr, 0, Term::Private::to_wide(key).c_str());
00019   std::wstring ret;
00020   if(size == 0 || size > ret.max_size()) return {false, std::string()};
00021   ret.reserve(size);
00022   _wgetenv_s(&size, &ret[0], size, Term::Private::to_wide(key).c_str());
00023   return {true, Term::Private::to_narrow(ret)};
00024 #else
00025   if(std::getenv(key.c_str()) != nullptr) { return {true,
      static_cast<std::string>(std::getenv(key.c_str()))}; }
00026   return {false, std::string()};
00027 #endif
00028 }
```

9.67 cpp-terminal/private/env.hpp File Reference

```
#include <string>
#include <utility>
```

Namespaces

- namespace Term
- namespace Term::Private

Functions

- std::pair< bool, std::string > Term::Private::getenv (const std::string &env)

  Value of an environment variables.

9.68 env.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 // This header should be used only in files contained inside platforms folder !!!
00011 #pragma once
00012
00013 #include <string>
00014 #include <utility>
00015
00016 namespace Term
00017 {
00018
00019 namespace Private
00020 {
00021
00029 std::pair<bool, std::string> getenv(const std::string& env);
00030
00031 }  // namespace Private
00032
00033 }  // namespace Term
```

9.69 cpp-terminal/private/exception.cpp File Reference

```
#include "cpp-terminal/private/exception.hpp"
#include "cpp-terminal/exception.hpp"
#include "cpp-terminal/version.hpp"
#include "return_code.hpp"
#include <atomic>
#include <exception>
#include "cpp-terminal/private/unicode.hpp"
#include <memory>
#include <windows.h>
#include <cerrno>
#include <cstdio>
#include <string>
```

9.70 exception.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/exception.hpp"
00011
00012 #include "cpp-terminal/exception.hpp"
00013 #include "cpp-terminal/version.hpp"
00014 #include "return_code.hpp"
00015
00016 #include <atomic>
00017 #include <exception>
00018
00019 #if defined(_WIN32)
00020   #include "cpp-terminal/private/unicode.hpp"
00021
00022   #include <memory>
00023   #include <windows.h>
00024   #if defined(MessageBox)
00025     #undef MessageBox
00026   #endif
00027 #else
00028   #include <cstring>
00029 #endif
00030
00031 #include <cerrno>
00032 #include <cstdio>
00033 #include <string>
00034
00035 Term::Exception::Exception(const std::string& message) noexcept : m_message(message) {}
00036
00037 Term::Exception::Exception(const std::int64_t& code, const std::string& message) noexcept :
      m_code(code), m_message(message) {}
00038
00039 const char* Term::Exception::what() const noexcept
00040 {
00041   build_what();
00042   return m_what.c_str();
00043 }
00044
00045 std::int64_t Term::Exception::code() const noexcept { return m_code; }
00046
00047 std::string Term::Exception::message() const noexcept { return m_message; }
00048
00049 std::string Term::Exception::context() const noexcept { return m_context; }
00050
00051 Term::Exception::Exception(const std::int64_t& code) noexcept : m_code(code) {}
00052
00053 void Term::Exception::build_what() const noexcept
00054 {
```

```
00055   if(0 == m_code) { m_what = m_message; }
00056   else { m_what = "error " + std::to_string(m_code) + ": " + m_message; }
00057 }
00058
00059 void Term::Exception::setMessage(const std::string& message) noexcept { m_message = message; }
00060
00061 void Term::Exception::setContext(const std::string& context) noexcept { m_context = context; }
00062
00063 void Term::Exception::setWhat(const std::string& what) const noexcept { m_what = what; }
00064
00065 #if defined(_WIN32)
00066
00067 std::int64_t Term::Private::WindowsError::error() const noexcept { return m_error; }
00068
00069 bool Term::Private::WindowsError::check_value() const noexcept { return m_check_value; }
00070
00071 Term::Private::WindowsError& Term::Private::WindowsError::check_if(const bool& ret) noexcept
00072 {
00073   m_error       = static_cast<std::int64_t>(GetLastError());
00074   m_check_value = ret;
00075   return *this;
00076 }
00077
00078 void Term::Private::WindowsError::throw_exception(const std::string& str) const
00079 {
00080   if(m_check_value) { throw Term::Private::WindowsException(m_error, str); }
00081 }
00082
00083 Term::Private::WindowsException::WindowsException(const std::int64_t& error, const std::string&
      context) : Term::Exception(static_cast<std::int64_t>(error))
00084 {
00085   setContext(context);
00086   wchar_t*    ptr{nullptr};
00087   const DWORD cchMsg{FormatMessageW(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS |
      FORMAT_MESSAGE_ALLOCATE_BUFFER, nullptr, static_cast<uint32_t>(code()), MAKELANGID(LANG_ENGLISH,
      SUBLANG_ENGLISH_US), reinterpret_cast<wchar_t*>(&ptr), 0, nullptr)};
00088   if(cchMsg > 0)
00089   {
00090     auto deleter = [](void* p)
00091     {
00092       if(p != nullptr) { ::LocalFree(p); }
00093     };
00094     std::unique_ptr<wchar_t, decltype(deleter)> ptrBuffer(ptr, deleter);
00095     std::string                                 ret{Term::Private::to_narrow(ptrBuffer.get())};
00096     if(ret.size() >= 2 && ret[ret.size() - 1] == '\n' && ret[ret.size() - 2] == '\r')
      ret.erase(ret.size() - 2);
00097     setMessage(ret);
00098   }
00099   else { throw Term::Exception(::GetLastError(), "Error in FormatMessageW"); }
00100 }
00101
00102 void Term::Private::WindowsException::build_what() const noexcept
00103 {
00104   std::string what{std::string("windows error ") + std::to_string(code()) + std::string(": ") +
      message().c_str()};
00105   if(!context().empty()) what += " [" + context() + "]";
00106   setWhat(what);
00107 }
00108
00109 #endif
00110
00111 Term::Private::Errno::Errno() noexcept
00112 {
00113 #if defined(_WIN32)
00114   _set_errno(0);
00115 #else
00116   errno = {0};  //NOSONAR
00117 #endif
00118 }
00119
00120 Term::Private::Errno::~Errno() noexcept
00121 {
00122 #if defined(_WIN32)
00123   _set_errno(0);
00124 #else
00125   errno = {0};  //NOSONAR
00126 #endif
00127 }
00128
00129 std::int64_t Term::Private::Errno::error() const noexcept { return m_errno; }
00130
00131 bool Term::Private::Errno::check_value() const noexcept { return m_check_value; }
00132
00133 Term::Private::Errno& Term::Private::Errno::check_if(const bool& ret) noexcept
00134 {
00135 #if defined(_WIN32)
00136   int err{static_cast<int>(m_errno)};
```

```
00137   _get_errno(&err);
00138 #else
00139   m_errno = static_cast<std::uint32_t>(errno);  //NOSONAR
00140 #endif
00141   m_check_value = {ret};
00142   return *this;
00143 }
00144
00145 void Term::Private::Errno::throw_exception(const std::string& str) const
00146 {
00147   if(m_check_value) { throw Term::Private::ErrnoException(m_errno, str); }
00148 }
00149
00150 Term::Private::ErrnoException::ErrnoException(const std::int64_t& error, const std::string& context) :
      Term::Exception(error)
00151 {
00152   setContext(context);
00153 #if defined(_WIN32)
00154   std::wstring message(m_maxSize, L'\0');
00155   message = _wcserror_s(&message[0], message.size(), static_cast<int>(error));
00156   setMessage(Term::Private::to_narrow(message.c_str()));
00157 #else
00158   std::string message(m_maxSize, '\0');
00159   message = ::strerror_r(static_cast<std::int32_t>(error), &message[0], message.size());  //
      NOLINT(readability-container-data-pointer)
00160   setMessage(message);
00161 #endif
00162 }
00163
00164 void Term::Private::ErrnoException::build_what() const noexcept
00165 {
00166   std::string what{"errno " + std::to_string(code()) + ": " + message()};
00167   if(!context().empty()) { what += +" [" + context() + "]"; }
00168   setWhat(what);
00169 }
00170
00171 void Term::Private::ExceptionHandler(const ExceptionDestination& destination) noexcept
00172 {
00173   try
00174   {
00175     std::exception_ptr exception{std::current_exception()};
00176     if(exception != nullptr) { std::rethrow_exception(exception); }
00177   }
00178   catch(const Term::Exception& exception)
00179   {
00180     switch(destination)
00181     {
00182       case ExceptionDestination::MessageBox:
00183 #if defined(_WIN32)
00184         MessageBoxW(nullptr, Term::Private::to_wide(exception.what()).c_str(),
      Term::Private::to_wide("cpp-terminal v" + Term::Version::string()).c_str(), MB_OK | MB_ICONERROR);
00185         break;
00186 #endif
00187       case ExceptionDestination::StdErr:
00188 #if defined(_WIN32)
00189         (void)(fputws(Term::Private::to_wide(std::string("cpp-terminal v" + Term::Version::string() +
      "\n" + exception.what() + "\n")).c_str(), stderr));
00190 #else
00191         (void)(fputs(std::string("cpp-terminal v" + Term::Version::string() + "\n" + exception.what()
      + "\n").c_str(), stderr));
00192 #endif
00193         break;
00194       default: break;
00195     }
00196   }
00197   catch(const std::exception& exception)
00198   {
00199     switch(destination)
00200     {
00201       case ExceptionDestination::MessageBox:
00202 #if defined(_WIN32)
00203         MessageBoxW(nullptr, Term::Private::to_wide(exception.what()).c_str(),
      Term::Private::to_wide("cpp-terminal v" + Term::Version::string()).c_str(), MB_OK | MB_ICONERROR);
00204         break;
00205 #endif
00206       case ExceptionDestination::StdErr:
00207 #if defined(_WIN32)
00208         (void)(fputws(Term::Private::to_wide(std::string("cpp-terminal v" + Term::Version::string() +
      "\n" + exception.what() + "\n")).c_str(), stderr));
00209 #else
00210         (void)(fputs(std::string("cpp-terminal v" + Term::Version::string() + "\n" + exception.what()
      + "\n").c_str(), stderr));
00211 #endif
00212         break;
00213       default: break;
00214     }
00215   }
```

```
00216   catch(...)
00217   {
00218     switch(destination)
00219     {
00220       case ExceptionDestination::MessageBox:
00221 #if defined(_WIN32)
00222         MessageBoxW(nullptr, Term::Private::to_wide("cpp-terminal v" + Term::Version::string() +
      "Unknown error").c_str(), Term::Private::to_wide("cpp-terminal v" + Term::Version::string()).c_str(),
      MB_OK | MB_ICONERROR);
00223         break;
00224 #endif
00225       case ExceptionDestination::StdErr:
00226 #if defined(_WIN32)
00227         (void)(fputws(Term::Private::to_wide("cpp-terminal v" + Term::Version::string() + ": Unknown
      error\n").c_str(), stderr));
00228 #else
00229         (void)(fputs(("cpp-terminal v" + Term::Version::string() + ": Unknown error\n").c_str(),
      stderr));
00230 #endif
00231       default: break;
00232     }
00233   }
00234   (void)(std::fflush(stderr));
00235   std::_Exit(Term::returnCode());
00236 }
```

9.71 cpp-terminal/private/file.cpp File Reference

```
#include "cpp-terminal/private/file.hpp"
#include "cpp-terminal/private/exception.hpp"
#include <cstdio>
#include <new>
#include <io.h>
#include <windows.h>
#include "cpp-terminal/private/unicode.hpp"
#include <array>
#include <fcntl.h>
```

9.72 file.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/file.hpp"
00011
00012 #include "cpp-terminal/private/exception.hpp"
00013
00014 #include <cstdio>
00015 #include <new>
00016
00017 #if defined(_WIN32)
00018   #include <io.h>
00019   #include <windows.h>
00020 #else
00021   #include <sys/ioctl.h>
00022   #include <unistd.h>
00023 #endif
00024
00025 #include "cpp-terminal/private/unicode.hpp"
00026
00027 #include <array>
00028 #include <fcntl.h>
00029
00030 //FIXME Move this to other file
00031
```

```
00032 namespace
00033 {
00034 std::array<char, sizeof(Term::Private::InputFileHandler)>  stdin_buffer;
      //NOLINT(fuchsia-statically-constructed-objects)
00035 std::array<char, sizeof(Term::Private::OutputFileHandler)> stdout_buffer;
      //NOLINT(fuchsia-statically-constructed-objects)
00036 }  // namespace
00037
00038 Term::Private::InputFileHandler&  Term::Private::in  =
      reinterpret_cast<Term::Private::InputFileHandler&>(stdin_buffer);
00039 Term::Private::OutputFileHandler& Term::Private::out =
      reinterpret_cast<Term::Private::OutputFileHandler&>(stdout_buffer);
00040
00041 //
00042
00043 Term::Private::FileHandler::FileHandler(std::recursive_mutex& mutex, const std::string& file, const
      std::string& mode) noexcept
00044 try : m_mutex(mutex)
00045 {
00046 #if defined(_WIN32)
00047   m_handle = {CreateFile(file.c_str(), GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ |
      FILE_SHARE_WRITE, nullptr, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, nullptr)};
00048   if(m_handle == INVALID_HANDLE_VALUE)
00049   {
00050     Term::Private::WindowsError().check_if((m_handle = CreateFile("NUL", GENERIC_READ | GENERIC_WRITE,
      FILE_SHARE_READ | FILE_SHARE_WRITE, nullptr, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, nullptr)) ==
      INVALID_HANDLE_VALUE).throw_exception("Problem opening NUL");
00051     m_null = true;
00052   }
00053   Term::Private::Errno().check_if((m_fd = _open_osfhandle(reinterpret_cast<intptr_t>(m_handle),
      _O_RDWR)) == -1).throw_exception("_open_osfhandle(reinterpret_cast<intptr_t>(m_handle), _O_RDWR)");
00054   Term::Private::Errno().check_if(nullptr == (m_file = _fdopen(m_fd,
      mode.c_str()))).throw_exception("_fdopen(m_fd, mode.c_str())");
00055 #else
00056   std::size_t flag{O_ASYNC | O_DSYNC | O_NOCTTY | O_SYNC | O_NDELAY};
00057   flag &= ~static_cast<std::size_t>(O_NONBLOCK);
00058   if(mode.find('r') != std::string::npos) { flag |= O_RDONLY; }
      //NOLINT(abseil-string-find-str-contains)
00059   else if(mode.find('w') != std::string::npos) { flag |= O_WRONLY; }
      //NOLINT(abseil-string-find-str-contains)
00060   else { flag |= O_RDWR; }
00061   m_fd = {::open(file.c_str(), static_cast<int>(flag))};
      //NOLINT(cppcoreguidelines-pro-type-vararg,hicpp-vararg)
00062   if(m_fd == -1)
00063   {
00064     Term::Private::Errno().check_if((m_fd = ::open("/dev/null", static_cast<int>(flag))) ==
      -1).throw_exception(R"(::open("/dev/null", flag))");
      //NOLINT(cppcoreguidelines-pro-type-vararg,hicpp-vararg)
00065     m_null = true;
00066   }
00067   Term::Private::Errno().check_if(nullptr == (m_file = ::fdopen(m_fd,
      mode.c_str()))).throw_exception("::fdopen(m_fd, mode.c_str())");
00068   m_handle = m_file;
00069 #endif
00070   Term::Private::Errno().check_if(std::setvbuf(m_file, nullptr, _IONBF, 0) !=
      0).throw_exception("std::setvbuf(m_file, nullptr, _IONBF, 0)");
00071 }
00072 catch(...)
00073 {
00074   ExceptionHandler(ExceptionDestination::StdErr);
00075 }
00076
00077 Term::Private::FileHandler::~FileHandler() noexcept
00078 try
00079 {
00080   flush();
00081   Term::Private::Errno().check_if(0 != std::fclose(m_file)).throw_exception("std::fclose(m_file)");
      //NOLINT(cppcoreguidelines-owning-memory)
00082 }
00083 catch(...)
00084 {
00085   ExceptionHandler(ExceptionDestination::StdErr);
00086 }
00087
00088 bool Term::Private::FileHandler::null() const noexcept { return m_null; }
00089
00090 std::FILE* Term::Private::FileHandler::file() const noexcept { return m_file; }
00091
00092 std::int32_t Term::Private::FileHandler::fd() const noexcept { return m_fd; }
00093
00094 Term::Private::FileHandler::Handle Term::Private::FileHandler::handle() const noexcept { return
      m_handle; }
00095
00096 std::size_t Term::Private::OutputFileHandler::write(const std::string& str) const
00097 {
00098 #if defined(_WIN32)
00099   DWORD written{0};
```

```
00100   if(!str.empty()) { Term::Private::WindowsError().check_if(0 == WriteConsole(handle(), &str[0],
      static_cast<DWORD>(str.size()), &written, nullptr)).throw_exception("WriteConsole(handle(), &str[0],
      static_cast<DWORD>(str.size()), &written, nullptr)"); }
00101   return static_cast<std::size_t>(written);
00102 #else
00103   ssize_t written{0};
00104   if(!str.empty()) { Term::Private::Errno().check_if((written = ::write(fd(), str.data(), str.size()))
      == -1).throw_exception("::write(fd(), str.data(), str.size())"); }
00105   return static_cast<std::size_t>(written);
00106 #endif
00107 }
00108
00109 std::size_t Term::Private::OutputFileHandler::write(const char& character) const
00110 {
00111 #if defined(_WIN32)
00112   DWORD written{0};
00113   Term::Private::WindowsError().check_if(0 == WriteConsole(handle(), &character, 1, &written,
      nullptr)).throw_exception("WriteConsole(handle(), &character, 1, &written, nullptr)");
00114   return static_cast<std::size_t>(written);
00115 #else
00116   ssize_t written{0};
00117   Term::Private::Errno().check_if((written = ::write(fd(), &character, 1)) ==
      -1).throw_exception("::write(fd(), &character, 1)");
00118   return static_cast<std::size_t>(written);
00119 #endif
00120 }
00121
00122 std::string Term::Private::InputFileHandler::read() const
00123 {
00124 #if defined(_WIN32)
00125   DWORD       nread{0};
00126   std::string ret(4096, '\0');
00127   errno = 0;
00128   ReadConsole(Private::in.handle(), &ret[0], static_cast<DWORD>(ret.size()), &nread, nullptr);
00129   return ret.c_str();
00130 #else
00131   std::size_t nread{0};
00132   Term::Private::Errno().check_if(::ioctl(Private::in.fd(), FIONREAD, &nread) !=
      0).throw_exception("::ioctl(Private::in.fd(), FIONREAD, &nread)");
      //NOLINT(cppcoreguidelines-pro-type-vararg,hicpp-vararg)
00133   std::string ret(nread, '\0');
00134   if(nread != 0)
00135   {
00136     Term::Private::Errno().check_if(::read(Private::in.fd(), &ret[0], ret.size()) ==
      -1).throw_exception("::read(Private::in.fd(), &ret[0], ret.size())");
      //NOLINT(readability-container-data-pointer)
00137   }
00138   return ret;
00139 #endif
00140 }
00141
00142 void Term::Private::FileHandler::flush() { Term::Private::Errno().check_if(0 !=
      std::fflush(m_file)).throw_exception("std::fflush(m_file)"); }
00143
00144 void Term::Private::FileHandler::lockIO() { m_mutex.lock(); }
00145 void Term::Private::FileHandler::unlockIO() { m_mutex.unlock(); }
00146
00147 Term::Private::OutputFileHandler::OutputFileHandler(std::recursive_mutex& io_mutex) noexcept
00148 try : FileHandler(io_mutex, m_file, "w")
00149 {
00150   //noop
00151 }
00152 catch(...)
00153 {
00154   ExceptionHandler(ExceptionDestination::StdErr);
00155 }
00156
00157 Term::Private::InputFileHandler::InputFileHandler(std::recursive_mutex& io_mutex) noexcept
00158 try : FileHandler(io_mutex, m_file, "r")
00159 {
00160   //noop
00161 }
00162 catch(...)
00163 {
00164   ExceptionHandler(ExceptionDestination::StdErr);
00165 }
00166
00167 std::string Term::Private::ask(const std::string& str)
00168 {
00169   Term::Private::out.write(str);
00170   std::string ret{Term::Private::in.read()};
00171   for(std::size_t i = 0; i != ret.size(); ++i) { Term::Private::out.write("\b \b"); }
00172   return ret;
00173 }
```

9.73 cpp-terminal/private/file.hpp File Reference

```
#include "cpp-terminal/private/file_initializer.hpp"
#include <cstddef>
#include <cstdint>
#include <mutex>
#include <string>
```

Classes

- class Term::Private::FileHandler
- class Term::Private::OutputFileHandler
- class Term::Private::InputFileHandler

Namespaces

- namespace Term
- namespace Term::Private

Functions

- std::string Term::Private::ask (const std::string &str)

Variables

- InputFileHandler & Term::Private::in = reinterpret_cast<Term::Private::InputFileHandler&>(stdin_buffer)
- OutputFileHandler & Term::Private::out = reinterpret_cast<Term::Private::OutputFileHandler&>(stdout_↩ buffer)

9.74 file.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/private/file_initializer.hpp"
00013
00014 #include <cstddef>
00015 #include <cstdint>
00016 #include <mutex>
00017 #include <string>
00018
00019 namespace Term
00020 {
00021
00022 namespace Private
00023 {
00024
00025 class FileHandler
00026 {
00027 public:
```

```
00028 #if defined(_WIN32)
00029   using Handle = void*;
00030 #else
00031   using Handle = std::FILE*;
00032 #endif
00033   FileHandler(std::recursive_mutex& mutex, const std::string& file, const std::string& mode) noexcept;
00034   FileHandler(const FileHandler&)            = delete;
00035   FileHandler(FileHandler&&)                 = delete;
00036   FileHandler& operator=(const FileHandler&) = delete;
00037   FileHandler& operator=(FileHandler&&)      = delete;
00038   virtual ~FileHandler() noexcept;
00039   Handle       handle() const noexcept;
00040   bool         null() const noexcept;
00041   std::FILE*   file() const noexcept;
00042   std::int32_t fd() const noexcept;
00043   void         lockIO();
00044   void         unlockIO();
00045   void         flush();
00046
00047 private:
00048   // should be static but MacOS don't want it (crash at runtime)
00049   std::recursive_mutex& m_mutex;  //NOLINT(cppcoreguidelines-avoid-const-or-ref-data-members)
00050   bool                  m_null{false};
00051   Handle                m_handle{nullptr};
00052   FILE*                 m_file{nullptr};
00053   std::int32_t          m_fd{-1};
00054 };
00055
00056 class OutputFileHandler : public FileHandler
00057 {
00058 public:
00059   explicit OutputFileHandler(std::recursive_mutex& io_mutex) noexcept;
00060   OutputFileHandler(const OutputFileHandler& other)          = delete;
00061   OutputFileHandler(OutputFileHandler&& other)               = delete;
00062   OutputFileHandler& operator=(OutputFileHandler&& rhs)      = delete;
00063   OutputFileHandler& operator=(const OutputFileHandler& rhs) = delete;
00064   ~OutputFileHandler() override                              = default;
00065
00066   std::size_t write(const std::string& str) const;
00067   std::size_t write(const char& character) const;
00068 #if defined(_WIN32)
00069   static const constexpr char* m_file{"CONOUT$"};
00070 #else
00071   static const constexpr char* m_file{"/dev/tty"};
00072 #endif
00073 };
00074
00075 class InputFileHandler : public FileHandler
00076 {
00077 public:
00078   explicit InputFileHandler(std::recursive_mutex& io_mutex) noexcept;
00079   InputFileHandler(const InputFileHandler&)            = delete;
00080   InputFileHandler(InputFileHandler&&)                 = delete;
00081   InputFileHandler& operator=(InputFileHandler&&)      = delete;
00082   InputFileHandler& operator=(const InputFileHandler&) = delete;
00083   ~InputFileHandler() override                         = default;
00084
00085   std::string read() const;
00086 #if defined(_WIN32)
00087   static const constexpr char* m_file{"CONIN$"};
00088 #else
00089   static const constexpr char* m_file{"/dev/tty"};
00090 #endif
00091 };
00092
00093 extern InputFileHandler&     in;
00094 extern OutputFileHandler&    out;
00095 static const FileInitializer file_initializer;
00096
00097 std::string ask(const std::string& str);
00098
00099 }  // namespace Private
00100
00101 }  // namespace Term
```

9.75 cpp-terminal/private/file_initializer.cpp File Reference

```
#include "cpp-terminal/private/file_initializer.hpp"
#include "cpp-terminal/private/exception.hpp"
#include "cpp-terminal/private/file.hpp"
#include "cpp-terminal/private/unicode.hpp"
```

```
#include <io.h>
#include <windows.h>
```

9.76 file_initializer.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/file_initializer.hpp"
00011
00012 #include "cpp-terminal/private/exception.hpp"
00013 #include "cpp-terminal/private/file.hpp"
00014
00015 #if defined(_WIN32)
00016   #include "cpp-terminal/private/unicode.hpp"
00017
00018   #include <io.h>
00019   #include <windows.h>
00020   #if defined(MessageBox)
00021     #undef MessageBox
00022   #endif
00023 #else
00024   #include <sys/stat.h>
00025 #endif
00026
00027 bool Term::Private::FileInitializer::m_consoleCreated = {false};
00028
00029 std::size_t Term::Private::FileInitializer::m_counter = {0};
00030
00031 void Term::Private::FileInitializer::attachConsole() noexcept
00032 try
00033 {
00034 #if defined(_WIN32)
00035   // If something happen here we still don't have a console so we can only use a MessageBox to warn
      the users something is very bad and that they should contact us.
00036   Term::Private::WindowsError
      error{Term::Private::WindowsError().check_if(AttachConsole(ATTACH_PARENT_PROCESS) == 0)};
00037   bool                        need_allocation{false};
00038   switch(error.error())
00039   {
00040     case ERROR_SUCCESS: break;
      // Has been attached
00041     case ERROR_ACCESS_DENIED: need_allocation = false; break;
      // Already attached that's good !
00042     case ERROR_INVALID_PARAMETER: error.throw_exception("The specified process does not exist !");
      break;  // Should never happen !
00043     case ERROR_INVALID_HANDLE: need_allocation = true; break;
      // Need to allocate th console !
00044   }
00045   if(need_allocation)
00046   {
00047     Term::Private::WindowsError().check_if(AllocConsole() == 0).throw_exception("AllocConsole()");
00048     m_consoleCreated = true;
00049   }
00050 #endif
00051 }
00052 catch(...)
00053 {
00054   detachConsole();
00055   ExceptionHandler(ExceptionDestination::MessageBox);
00056 }
00057
00058 void Term::Private::FileInitializer::detachConsole() noexcept
00059 try
00060 {
00061 #if defined(_WIN32)
00062   if(m_consoleCreated) { Term::Private::WindowsError().check_if(0 ==
      FreeConsole()).throw_exception("FreeConsole()"); }
00063 #endif
00064 }
00065 catch(...)
00066 {
00067   ExceptionHandler(ExceptionDestination::MessageBox);
```

```
00068 }
00069
00070 Term::Private::FileInitializer::FileInitializer() noexcept
00071 try
00072 {
00073   // MacOS was not happy wish a static mutex in the class so we create it and pass to each class;
00074   static std::recursive_mutex ioMutex;
00075   if(0 == m_counter)
00076   {
00077     attachConsole();
00078     openStandardStreams();
00079     if(nullptr == new(&Term::Private::in) InputFileHandler(ioMutex)) { throw
      Term::Exception("new(&Term::Private::in) InputFileHandler(ioMutex)"); }
00080     if(nullptr == new(&Term::Private::out) OutputFileHandler(ioMutex)) { throw
      Term::Exception("new(&Term::Private::out) OutputFileHandler(ioMutex)"); }
00081   }
00082   ++m_counter;
00083 }
00084 catch(...)
00085 {
00086   ExceptionHandler(ExceptionDestination::StdErr);
00087 }
00088
00089 Term::Private::FileInitializer::~FileInitializer() noexcept
00090 try
00091 {
00092   --m_counter;
00093   if(0 == m_counter)
00094   {
00095     (&Term::Private::in)->~InputFileHandler();    //NOSONAR(S3432)
00096     (&Term::Private::out)->~OutputFileHandler();  //NOSONAR(S3432)
00097     detachConsole();
00098   }
00099 }
00100 catch(...)
00101 {
00102   ExceptionHandler(ExceptionDestination::StdErr);
00103 }
00104
00105 void Term::Private::FileInitializer::openStandardStreams() noexcept
00106 try
00107 {
00108 #if defined(_WIN32)
00109   FILE* fDummy{nullptr};
00110   if(_fileno(stderr) < 0 || _get_osfhandle(_fileno(stderr)) < 0) {
      Term::Private::Errno().check_if(_wfreopen_s(&fDummy, L"CONOUT$", L"w", stderr) !=
      0).throw_exception(R"(_wfreopen_s(&fDummy, L"CONOUT$", L"w", stderr))"); }
00111   if(_fileno(stdout) < 0 || _get_osfhandle(_fileno(stdout)) < 0) {
      Term::Private::Errno().check_if(_wfreopen_s(&fDummy, L"CONOUT$", L"w", stdout) !=
      0).throw_exception(R"(_wfreopen_s(&fDummy, L"CONOUT$", L"w", stdout))"); }
00112   if(_fileno(stdin) < 0 || _get_osfhandle(_fileno(stdin)) < 0) {
      Term::Private::Errno().check_if(_wfreopen_s(&fDummy, L"CONIN$", L"r", stdin) !=
      0).throw_exception(R"(_wfreopen_s(&fDummy, L"CONIN$", L"r", stdin))"); }
00113   const std::size_t bestSize{BUFSIZ > 4096 ? BUFSIZ : 4096};
00114 #else
00115   if(::fileno(stderr) < 0) { Term::Private::Errno().check_if(nullptr == std::freopen("/dev/tty", "w",
      stderr)).throw_exception(R"(std::freopen("/dev/tty", "w", stderr))"); }
      //NOLINT(cppcoreguidelines-owning-memory)
00116   if(::fileno(stdout) < 0) { Term::Private::Errno().check_if(nullptr == std::freopen("/dev/tty", "w",
      stdout)).throw_exception(R"(std::freopen("/dev/tty", "w", stdout))"); }
      //NOLINT(cppcoreguidelines-owning-memory)
00117   if(::fileno(stdin) < 0) { Term::Private::Errno().check_if(nullptr == std::freopen("/dev/tty", "r",
      stdin)).throw_exception(R"(std::freopen("/dev/tty", "r", stdin))"); }
      //NOLINT(cppcoreguidelines-owning-memory)
00118   struct stat stats = {};
00119   ::stat("/dev/tty", &stats);
00120   const std::size_t bestSize{static_cast<std::size_t>(stats.st_blksize) > 0 ?
      static_cast<std::size_t>(stats.st_blksize) : BUFSIZ};  //NOSONAR(S1774)
00121 #endif
00122   Term::Private::Errno().check_if(std::setvbuf(stderr, nullptr, _IONBF, 0) !=
      0).throw_exception("std::setvbuf(stderr, nullptr, _IONBF, 0)");
00123   Term::Private::Errno().check_if(std::setvbuf(stdout, nullptr, _IOLBF, bestSize) !=
      0).throw_exception("std::setvbuf(stdout, nullptr, _IOLBF, bestSize)");
00124   Term::Private::Errno().check_if(std::setvbuf(stdin, nullptr, _IOLBF, bestSize) !=
      0).throw_exception("std::setvbuf(stdin, nullptr, _IOLBF, bestSize)");
00125 }
00126 catch(...)
00127 {
00128   ExceptionHandler(ExceptionDestination::StdErr);
00129 }
```

9.77 cpp-terminal/private/file_initializer.hpp File Reference

`#include <cstddef>`

Classes

- class Term::Private::FileInitializer

Namespaces

- namespace Term
- namespace Term::Private

9.78 file_initializer.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstddef>
00013
00014 namespace Term
00015 {
00016
00017 namespace Private
00018 {
00019
00020 class FileInitializer
00021 {
00022 public:
00023   ~FileInitializer() noexcept;
00024   FileInitializer() noexcept;
00025   FileInitializer(FileInitializer&&)                 = delete;
00026   FileInitializer(const FileInitializer&)            = delete;
00027   FileInitializer& operator=(const FileInitializer&) = delete;
00028   FileInitializer& operator=(FileInitializer&&)      = delete;
00029
00030 private:
00031   static bool        m_consoleCreated;
00032   static std::size_t m_counter;
00033
00039   static void attachConsole() noexcept;
00040
00046   static void detachConsole() noexcept;
00047
00053   static void openStandardStreams() noexcept;
00054 };
00055
00056 }  // namespace Private
00057
00058 }  // namespace Term
```

9.79 cpp-terminal/private/input.cpp File Reference

```
#include "cpp-terminal/private/unicode.hpp"
#include <vector>
#include <windows.h>
#include "cpp-terminal/event.hpp"
#include "cpp-terminal/exception.hpp"
#include "cpp-terminal/input.hpp"
#include "cpp-terminal/private/blocking_queue.hpp"
#include "cpp-terminal/private/file.hpp"
#include "cpp-terminal/private/input.hpp"
#include "cpp-terminal/private/sigwinch.hpp"
#include <mutex>
#include <string>
```

Functions

- Term::Button::Action getAction (const std::int32_t &old_state, const std::int32_t &state, const std::int32_t &type)
- Term::Button setButton (const std::int32_t &old_state, const std::int32_t &state)
- void sendString (Term::Private::BlockingQueue &events, std::wstring &str)

9.79.1 Function Documentation

getAction()

```
Term::Button::Action getAction (
            const std::int32_t & old_state,
            const std::int32_t & state,
            const std::int32_t & type )
```

Definition at line 38 of file input.cpp.

```
00039 {
00040   if(((state - old_state) » (type - 1)) == 1) return Term::Button::Action::Pressed;
00041   else if(((old_state - state) » (type - 1)) == 1)
00042     return Term::Button::Action::Released;
00043   else
00044     return Term::Button::Action::None;
00045 }
```

sendString()

```
void sendString (
            Term::Private::BlockingQueue & events,
            std::wstring & str )
```

Definition at line 67 of file input.cpp.

```
00068 {
00069   if(!str.empty())
00070   {
00071     events.push(Term::Event(Term::Private::to_narrow(str.c_str())));
00072     str.clear();
00073   }
00074 }
```

setButton()

```
Term::Button setButton (
            const std::int32_t & old_state,
            const std::int32_t & state )
```

Definition at line 46 of file input.cpp.

```
00047 {
00048   Term::Button::Action action;
00049   action = getAction(old_state, state, FROM_LEFT_1ST_BUTTON_PRESSED);
00050   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Left, action);
00051
00052   action = getAction(old_state, state, FROM_LEFT_2ND_BUTTON_PRESSED);
00053   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Button1, action);
00054
00055   action = getAction(old_state, state, FROM_LEFT_3RD_BUTTON_PRESSED);
00056   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Button2, action);
00057
00058   action = getAction(old_state, state, FROM_LEFT_4TH_BUTTON_PRESSED);
00059   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Button3, action);
00060
00061   action = getAction(old_state, state, RIGHTMOST_BUTTON_PRESSED);
00062   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Right, action);
00063
00064   return Term::Button(Term::Button::Type::None, Term::Button::Action::None);
00065 }
```

9.80 input.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #if defined(_WIN32)
00011   #include "cpp-terminal/private/unicode.hpp"
00012
00013   #include <vector>
00014   #include <windows.h>
00015 #elif defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00016   #include <cerrno>
00017   #include <csignal>
00018   #include <sys/ioctl.h>
00019   #include <thread>
00020   #include <unistd.h>
00021 #else
00022   #include <memory>
00023   #include <sys/epoll.h>
00024 #endif
00025
00026 #include "cpp-terminal/event.hpp"
00027 #include "cpp-terminal/exception.hpp"
00028 #include "cpp-terminal/input.hpp"
00029 #include "cpp-terminal/private/blocking_queue.hpp"
00030 #include "cpp-terminal/private/file.hpp"
00031 #include "cpp-terminal/private/input.hpp"
00032 #include "cpp-terminal/private/sigwinch.hpp"
00033
00034 #include <mutex>
00035 #include <string>
00036
00037 #if defined(_WIN32)
00038 Term::Button::Action getAction(const std::int32_t& old_state, const std::int32_t& state, const
      std::int32_t& type)
00039 {
00040   if(((state - old_state) » (type - 1)) == 1) return Term::Button::Action::Pressed;
00041   else if(((old_state - state) » (type - 1)) == 1)
00042     return Term::Button::Action::Released;
00043   else
00044     return Term::Button::Action::None;
00045 }
00046 Term::Button setButton(const std::int32_t& old_state, const std::int32_t& state)
00047 {
```

```
00048   Term::Button::Action action;
00049   action = getAction(old_state, state, FROM_LEFT_1ST_BUTTON_PRESSED);
00050   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Left, action);
00051
00052   action = getAction(old_state, state, FROM_LEFT_2ND_BUTTON_PRESSED);
00053   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Button1, action);
00054
00055   action = getAction(old_state, state, FROM_LEFT_3RD_BUTTON_PRESSED);
00056   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Button2, action);
00057
00058   action = getAction(old_state, state, FROM_LEFT_4TH_BUTTON_PRESSED);
00059   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Button3, action);
00060
00061   action = getAction(old_state, state, RIGHTMOST_BUTTON_PRESSED);
00062   if(action != Term::Button::Action::None) return Term::Button(Term::Button::Type::Right, action);
00063
00064   return Term::Button(Term::Button::Type::None, Term::Button::Action::None);
00065 }
00066
00067 void sendString(Term::Private::BlockingQueue& events, std::wstring& str)
00068 {
00069   if(!str.empty())
00070   {
00071     events.push(Term::Event(Term::Private::to_narrow(str.c_str())));
00072     str.clear();
00073   }
00074 }
00075
00076 #endif
00077
00078 std::thread Term::Private::Input::m_thread = std::thread(Term::Private::Input::read_event);
00079
00080 Term::Private::BlockingQueue Term::Private::Input::m_events;
00081
00082 int Term::Private::Input::m_poll{-1};
00083
00084 void Term::Private::Input::init_thread()
00085 {
00086   Term::Private::Sigwinch::unblockSigwinch();
00087 #if defined(__linux__)
00088   m_poll = {::epoll_create1(EPOLL_CLOEXEC)};
00089   ::epoll_event signal;
00090   signal.events  = {EPOLLIN};
00091   signal.data.fd = {Term::Private::Sigwinch::get()};
00092   ::epoll_ctl(m_poll, EPOLL_CTL_ADD, Term::Private::Sigwinch::get(), &signal);
00093   ::epoll_event input;
00094   input.events  = {EPOLLIN};
00095   input.data.fd = {Term::Private::in.fd()};
00096   ::epoll_ctl(m_poll, EPOLL_CTL_ADD, Term::Private::in.fd(), &input);
00097 #endif
00098 }
00099
00100 void Term::Private::Input::read_event()
00101 {
00102   init_thread();
00103   while(true)
00104   {
00105 #if defined(_WIN32)
00106     WaitForSingleObject(Term::Private::in.handle(), INFINITE);
00107     read_raw();
00108 #elif defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00109     if(Term::Private::Sigwinch::isSigwinch()) m_events.push(screen_size());
00110     read_raw();
00111 #else
00112     ::epoll_event ret;
00113     if(epoll_wait(m_poll, &ret, 1, -1) == 1)
00114     {
00115       if(Term::Private::Sigwinch::isSigwinch(ret.data.fd)) m_events.push(Term::Screen(screen_size()));
00116       else
00117         read_raw();
00118     }
00119 #endif
00120   }
00121 }
00122
00123 #if defined(_WIN32)
00124 void Term::Private::Input::read_windows_key(const std::uint16_t& virtual_key_code, const
    std::uint32_t& control_key_state, const std::size_t& occurrence)
00125 {
00126   // First check if we have ALT etc (CTRL is already done so skip it)
00127   Term::MetaKey toAdd{Term::MetaKey::Value::None};
00128   if(((control_key_state & LEFT_ALT_PRESSED) == LEFT_ALT_PRESSED) || ((control_key_state &
    RIGHT_ALT_PRESSED) == RIGHT_ALT_PRESSED)) toAdd += Term::MetaKey::Value::Alt;
00129   if(((control_key_state & LEFT_CTRL_PRESSED) == LEFT_CTRL_PRESSED) || ((control_key_state &
    RIGHT_CTRL_PRESSED) == RIGHT_CTRL_PRESSED)) toAdd += Term::MetaKey::Value::Ctrl;
00130
00131   switch(virtual_key_code)
```

```
00132   {
00133     case VK_CANCEL:  //??
00134     case VK_CLEAR:   //??
00135     case VK_SHIFT:
00136     case VK_CONTROL:
00137     case VK_MENU:
00138     case VK_PAUSE:  //??
00139     case VK_CAPITAL:
00140     case VK_KANA:  //??
00141     //case VK_HANGUL: // Same
00142     case VK_JUNJA:  // ?
00143     case VK_FINAL:  // ?
00144     case VK_HANJA:
00145     //case VK_KANJI: // Same
00146     case VK_CONVERT:     // ?
00147     case VK_NONCONVERT:  // ?
00148     case VK_ACCEPT:      // ?
00149     case VK_MODECHANGE:  // ?
00150       break;
00151     case VK_PRIOR: m_events.push(std::move(toAdd + Term::Key(Key::Value::PageUp)), occurrence); break;
00152     case VK_NEXT: m_events.push(std::move(toAdd + Term::Key(Key::Value::PageDown)), occurrence);
    break;
00153     case VK_END: m_events.push(std::move(toAdd + Term::Key(Key::Value::End)), occurrence); break;
00154     case VK_HOME: m_events.push(std::move(toAdd + Term::Key(Key::Value::Home)), occurrence); break;
00155     case VK_LEFT: m_events.push(std::move(toAdd + Term::Key(Key::Value::ArrowLeft)), occurrence);
    break;
00156     case VK_UP: m_events.push(std::move(toAdd + Term::Key(Key::Value::ArrowUp)), occurrence); break;
00157     case VK_RIGHT: m_events.push(std::move(toAdd + Term::Key(Key::Value::ArrowRight)), occurrence);
    break;
00158     case VK_DOWN: m_events.push(std::move(toAdd + Term::Key(Key::Value::ArrowDown)), occurrence);
    break;
00159     case VK_SELECT:   //?
00160     case VK_PRINT:    //?
00161     case VK_EXECUTE:  //?
00162       break;
00163     case VK_SNAPSHOT: m_events.push(std::move(toAdd + Term::Key(Key::Value::PrintScreen)),
    occurrence); break;
00164     case VK_INSERT: m_events.push(std::move(toAdd + Term::Key(Key::Value::Insert)), occurrence);
    break;
00165     case VK_DELETE: m_events.push(std::move(toAdd + Term::Key(Key::Value::Del)), occurrence); break;
00166     case VK_HELP:   //?
00167     case VK_LWIN:   //Maybe allow to detect Windows key Left and right
00168     case VK_RWIN:   //Maybe allow to detect Windows key Left and right
00169     case VK_APPS:   //?
00170     case VK_SLEEP:  //?
00171       break;
00172     case VK_F1: m_events.push(std::move(toAdd + Term::Key(Key::Value::F1)), occurrence); break;
00173     case VK_F2: m_events.push(std::move(toAdd + Term::Key(Key::Value::F2)), occurrence); break;
00174     case VK_F3: m_events.push(std::move(toAdd + Term::Key(Key::Value::F3)), occurrence); break;
00175     case VK_F4: m_events.push(std::move(toAdd + Term::Key(Key::Value::F4)), occurrence); break;
00176     case VK_F5: m_events.push(std::move(toAdd + Term::Key(Key::Value::F5)), occurrence); break;
00177     case VK_F6: m_events.push(std::move(toAdd + Term::Key(Key::Value::F6)), occurrence); break;
00178     case VK_F7: m_events.push(std::move(toAdd + Term::Key(Key::Value::F7)), occurrence); break;
00179     case VK_F8: m_events.push(std::move(toAdd + Term::Key(Key::Value::F8)), occurrence); break;
00180     case VK_F9: m_events.push(std::move(toAdd + Term::Key(Key::Value::F9)), occurrence); break;
00181     case VK_F10: m_events.push(std::move(toAdd + Term::Key(Key::Value::F10)), occurrence); break;
00182     case VK_F11: m_events.push(std::move(toAdd + Term::Key(Key::Value::F11)), occurrence); break;
00183     case VK_F12: m_events.push(std::move(toAdd + Term::Key(Key::Value::F12)), occurrence); break;
00184     case VK_F13: m_events.push(std::move(toAdd + Term::Key(Key::Value::F13)), occurrence); break;
00185     case VK_F14: m_events.push(std::move(toAdd + Term::Key(Key::Value::F14)), occurrence); break;
00186     case VK_F15: m_events.push(std::move(toAdd + Term::Key(Key::Value::F15)), occurrence); break;
00187     case VK_F16: m_events.push(std::move(toAdd + Term::Key(Key::Value::F16)), occurrence); break;
00188     case VK_F17: m_events.push(std::move(toAdd + Term::Key(Key::Value::F17)), occurrence); break;
00189     case VK_F18: m_events.push(std::move(toAdd + Term::Key(Key::Value::F18)), occurrence); break;
00190     case VK_F19: m_events.push(std::move(toAdd + Term::Key(Key::Value::F19)), occurrence); break;
00191     case VK_F20: m_events.push(std::move(toAdd + Term::Key(Key::Value::F20)), occurrence); break;
00192     case VK_F21: m_events.push(std::move(toAdd + Term::Key(Key::Value::F21)), occurrence); break;
00193     case VK_F22: m_events.push(std::move(toAdd + Term::Key(Key::Value::F22)), occurrence); break;
00194     case VK_F23: m_events.push(std::move(toAdd + Term::Key(Key::Value::F23)), occurrence); break;
00195     case VK_F24: m_events.push(std::move(toAdd + Term::Key(Key::Value::F24)), occurrence); break;
00196     case VK_NUMLOCK:
00197     case VK_SCROLL:
00198     default: break;
00199   }
00200 }
00201 #endif
00202
00203 void Term::Private::Input::read_raw()
00204 {
00205 #ifdef _WIN32
00206   DWORD to_read{0};
00207   GetNumberOfConsoleInputEvents(Private::in.handle(), &to_read);
00208   if(to_read == 0) return;
00209   DWORD                     read{0};
00210   std::vector<INPUT_RECORD> events{to_read};
00211   if(!ReadConsoleInputW(Private::in.handle(), &events[0], to_read, &read) || read != to_read)
    Term::Exception("ReadFile() failed");
```

```
00212   std::wstring ret;
00213   bool         need_windows_size{false};
00214   for(std::size_t i = 0; i != read; ++i)
00215   {
00216     switch(events[i].EventType)
00217     {
00218       case KEY_EVENT:
00219       {
00220         if(events[i].Event.KeyEvent.bKeyDown)
00221         {
00222           if(events[i].Event.KeyEvent.uChar.UnicodeChar == 0)
      read_windows_key(events[i].Event.KeyEvent.wVirtualKeyCode, events[i].Event.KeyEvent.dwControlKeyState,
      events[i].Event.KeyEvent.wRepeatCount);
00223           else
00224             ret.append(events[i].Event.KeyEvent.wRepeatCount,
      events[i].Event.KeyEvent.uChar.UnicodeChar == Term::Key::Del ?
      static_cast<wchar_t>(Key(Term::Key::Value::Backspace)) :
      static_cast<wchar_t>(events[i].Event.KeyEvent.uChar.UnicodeChar));
00225         }
00226         break;
00227       }
00228       case FOCUS_EVENT:
00229       {
00230         sendString(m_events, ret);
00231
      m_events.push(Event(Focus(static_cast<Term::Focus::Type>(events[i].Event.FocusEvent.bSetFocus))));
00232         break;
00233       }
00234       case MENU_EVENT:
00235       {
00236         sendString(m_events, ret);
00237         break;
00238       }
00239       case MOUSE_EVENT:
00240       {
00241         sendString(m_events, ret);
00242         static MOUSE_EVENT_RECORD old_state;
00243         if(events[i].Event.MouseEvent.dwEventFlags == MOUSE_WHEELED ||
      events[i].Event.MouseEvent.dwEventFlags == MOUSE_HWHEELED)
00244           ;
00245         else if(old_state.dwButtonState == events[i].Event.MouseEvent.dwButtonState &&
      old_state.dwMousePosition.X == events[i].Event.MouseEvent.dwMousePosition.X &&
      old_state.dwMousePosition.Y == events[i].Event.MouseEvent.dwMousePosition.Y && old_state.dwEventFlags
      == events[i].Event.MouseEvent.dwEventFlags)
00246           break;
00247         std::int32_t state{static_cast<std::int32_t>(events[i].Event.MouseEvent.dwButtonState)};
00248         switch(events[i].Event.MouseEvent.dwEventFlags)
00249         {
00250           case 0:
00251           {
00252             m_events.push(Term::Mouse(setButton(static_cast<std::int32_t>(old_state.dwButtonState),
      state), static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00253             ;
00254             break;
00255           }
00256           case MOUSE_MOVED:
00257           {
00258             m_events.push(Term::Mouse(setButton(static_cast<std::int32_t>(old_state.dwButtonState),
      state), static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00259             ;
00260             break;
00261           }
00262           case DOUBLE_CLICK:
00263           {
00264
      m_events.push(Term::Mouse(Term::Button(setButton(static_cast<std::int32_t>(old_state.dwButtonState),
      state).type(), Term::Button::Action::DoubleClicked),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00265             break;
00266           }
00267           case MOUSE_WHEELED:
00268           {
00269             if(state > 0) m_events.push(Term::Mouse(Button(Term::Button::Type::Wheel,
      Term::Button::Action::RolledUp),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00270             else
00271               m_events.push(Term::Mouse(Button(Term::Button::Type::Wheel,
      Term::Button::Action::RolledDown),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00272             break;
00273           }
00274           case MOUSE_HWHEELED:
```

```
00275           {
00276             if(state > 0) m_events.push(Term::Mouse(Button(Term::Button::Type::Wheel,
      Term::Button::Action::ToRight),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00277             else
00278               m_events.push(Term::Mouse(Button(Term::Button::Type::Wheel,
      Term::Button::Action::ToLeft),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.Y),
      static_cast<std::uint16_t>(events[i].Event.MouseEvent.dwMousePosition.X)));
00279             break;
00280           }
00281           default: break;
00282         }
00283         old_state = events[i].Event.MouseEvent;
00284         break;
00285       }
00286       case WINDOW_BUFFER_SIZE_EVENT:
00287       {
00288         need_windows_size = true;  // if we send directly it's too much generations
00289         sendString(m_events, ret);
00290         break;
00291       }
00292       default: break;
00293     }
00294   }
00295   sendString(m_events, ret);
00296   if(need_windows_size == true) { m_events.push(screen_size()); }
00297 #else
00298   Private::in.lockIO();
00299   std::string ret = Term::Private::in.read();
00300   Private::in.unlockIO();
00301   if(!ret.empty()) m_events.push(Event(ret.c_str()));
00302 #endif
00303 }
00304
00305 Term::Private::Input::Input() {}
00306
00307 void Term::Private::Input::startReading()
00308 {
00309   static bool activated{false};
00310   if(!activated)
00311   {
00312     m_thread.detach();
00313     activated = true;
00314   }
00315 }
00316
00317 Term::Event Term::Private::Input::getEvent() { return m_events.pop(); }
00318
00319 Term::Event Term::Private::Input::getEventBlocking()
00320 {
00321   static std::mutex                   cv_m;
00322   static std::unique_lock<std::mutex> lk(cv_m);
00323   if(m_events.empty()) m_events.wait_for_events(lk);
00324   return m_events.pop();
00325 }
00326
00327 static Term::Private::Input m_input;
00328
00329 Term::Event Term::read_event()
00330 {
00331   m_input.startReading();
00332   return m_input.getEventBlocking();
00333 }
```

9.81 cpp-terminal/private/macros.hpp File Reference

Macros

- #define CPP_TERMINAL_NODISCARD
- #define CPP_TERMINAL_FALLTHROUGH
- #define CPP_TERMINAL_MAYBE_UNUSED

9.81.1 Macro Definition Documentation

CPP_TERMINAL_FALLTHROUGH

```
#define CPP_TERMINAL_FALLTHROUGH
```

Definition at line 40 of file macros.hpp.

CPP_TERMINAL_MAYBE_UNUSED

```
#define CPP_TERMINAL_MAYBE_UNUSED
```

Definition at line 41 of file macros.hpp.

CPP_TERMINAL_NODISCARD

```
#define CPP_TERMINAL_NODISCARD
```

Definition at line 39 of file macros.hpp.

9.82 macros.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #if __cplusplus >= 201703L
00013
00014   #if defined(__GNUC__) && (__GNUC__ > 7)
00015     #define CPP_TERMINAL_NODISCARD [[nodiscard]]
00016   #elif defined(__clang_major__) && (__clang_major__ > 3 || (__clang_major__ == 3 && __clang_minor__ >
      8))
00017     #define CPP_TERMINAL_NODISCARD [[nodiscard]]
00018   #else
00019     #define CPP_TERMINAL_NODISCARD
00020   #endif
00021
00022   #if defined(__GNUC__) && (__GNUC__ > 5)
00023     #define CPP_TERMINAL_FALLTHROUGH [[fallthrough]]
00024   #elif defined(__clang_major__) && (__clang_major__ > 3 || (__clang_major__ == 3 && __clang_minor__ >
      5))
00025     #define CPP_TERMINAL_FALLTHROUGH [[fallthrough]]
00026   #else
00027     #define CPP_TERMINAL_FALLTHROUGH
00028   #endif
00029
00030   #if defined(__GNUC__) && (__GNUC__ > 5)
00031     #define CPP_TERMINAL_MAYBE_UNUSED [[maybe_unused]]
00032   #elif defined(__clang_major__) && (__clang_major__ > 3 || (__clang_major__ == 3 && __clang_minor__ >
      5))
00033     #define CPP_TERMINAL_MAYBE_UNUSED [[maybe_unused]]
00034   #else
00035     #define CPP_TERMINAL_MAYBE_UNUSED
00036   #endif
00037
00038 #else
00039   #define CPP_TERMINAL_NODISCARD
00040   #define CPP_TERMINAL_FALLTHROUGH
00041   #define CPP_TERMINAL_MAYBE_UNUSED
00042 #endif
```

9.83 cpp-terminal/private/README.md File Reference

9.84 README.md File Reference

9.85 cpp-terminal/private/return_code.cpp File Reference

```
#include "cpp-terminal/private/return_code.hpp"
#include "cpp-terminal/private/env.hpp"
#include <cstdlib>
#include <string>
#include <utility>
```

9.86 return_code.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/return_code.hpp"
00011
00012 #include "cpp-terminal/private/env.hpp"
00013
00014 #include <cstdlib>
00015 #include <string>
00016 #include <utility>
00017
00018 std::uint16_t Term::returnCode() noexcept
00019 {
00020   static std::uint16_t               code{EXIT_FAILURE};
00021   const std::pair<bool, std::string> returnCode{Private::getenv("CPP_TERMINAL_BADSTATE")};
00022   try
00023   {
00024     if(returnCode.first && (std::stoi(returnCode.second) != EXIT_SUCCESS)) { code =
      static_cast<std::uint16_t>(std::stoi(returnCode.second)); }
00025   }
00026   catch(...)
00027   {
00028     code = EXIT_FAILURE;
00029   }
00030   return code;
00031 }
```

9.87 cpp-terminal/private/return_code.hpp File Reference

```
#include <cstdint>
```

Namespaces

- namespace Term

Functions

- std::uint16_t Term::returnCode () noexcept

9.88 return_code.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstdint>
00013
00014 namespace Term
00015 {
00016
00017 std::uint16_t returnCode() noexcept;
00018
00019 }  // namespace Term
```

9.89 cpp-terminal/private/screen.cpp File Reference

```
#include "cpp-terminal/screen.hpp"
#include <windows.h>
#include "cpp-terminal/private/file.hpp"
```

9.90 screen.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/screen.hpp"
00011
00012 #ifdef _WIN32
00013   #include <windows.h>
00014 #else
00015   #include <sys/ioctl.h>
00016 #endif
00017
00018 #include "cpp-terminal/private/file.hpp"
00019
00020 Term::Screen Term::screen_size()
00021 {
00022 #ifdef _WIN32
00023   CONSOLE_SCREEN_BUFFER_INFO inf;
00024   if(GetConsoleScreenBufferInfo(Private::out.handle(), &inf)) return
      Term::Screen(static_cast<std::size_t>(inf.srWindow.Bottom - inf.srWindow.Top + 1),
      static_cast<std::size_t>(inf.srWindow.Right - inf.srWindow.Left + 1));
00025   return Term::Screen();
00026 #else
00027   Term::Screen ret;
00028   struct winsize window
00029   {
00030     0, 0, 0, 0
00031   };
00032   if(ioctl(Private::out.fd(), TIOCGWINSZ, &window) != -1) ret = {window.ws_row, window.ws_col};
00033   return ret;
00034 #endif
00035 }
```

9.91 cpp-terminal/screen.cpp File Reference

```
#include "cpp-terminal/screen.hpp"
```

9.92 screen.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/screen.hpp"
00011
00012 Term::Screen::Screen(const std::size_t& rows, const std::size_t& columns) : m_size({rows, columns}) {}
00013
00014 std::size_t Term::Screen::rows() const { return m_size.first; }
00015
00016 std::size_t Term::Screen::columns() const { return m_size.second; }
00017
00018 bool Term::Screen::empty() const { return (0 == m_size.second) && (0 == m_size.first); }
00019
00020 std::string Term::clear_screen() { return "\u001b[2J"; }
00021
00022 std::string Term::screen_save()
00023 {
00024   return "\u001b7\u001b[?1049h";  // save current cursor position, save screen FIXME
00025 }
00026
00027 std::string Term::screen_load()
00028 {
00029   return "\u001b[?1049l\u001b8";  // restores screen, restore current cursor position FIXME
00030 }
00031
00032 bool Term::Screen::operator==(const Term::Screen& screen) const { return (this->rows() ==
      screen.rows()) && (this->columns() == screen.columns()); }
00033
00034 bool Term::Screen::operator!=(const Term::Screen& screen) const { return !(*this == screen); }
```

9.93 cpp-terminal/private/sigwinch.cpp File Reference

```
#include "cpp-terminal/private/sigwinch.hpp"
#include "cpp-terminal/private/exception.hpp"
#include <sys/signalfd.h>
```

Namespaces

- namespace Term
- namespace Term::Private

Variables

- volatile std::sig_atomic_t Term::Private::m_signalStatus {0}

9.94 sigwinch.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/sigwinch.hpp"
00011
00012 #include "cpp-terminal/private/exception.hpp"
00013
00014 #if !defined(_WIN32)
00015   #include <csignal>
00016   #include <unistd.h>
00017 #endif
00018
00019 #if defined(__linux__)
00020   #include <sys/signalfd.h>
00021 #endif
00022
00023 #if defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00024 namespace Term
00025 {
00026 namespace Private
00027 {
00028 volatile std::sig_atomic_t m_signalStatus{0};
00029 static void                sigwinchHandler(int sig)
00030 {
00031   if(sig == SIGWINCH) { m_signalStatus = 1; }
00032   else { m_signalStatus = 0; }
00033 }
00034 }  // namespace Private
00035 }  // namespace Term
00036 #endif
00037
00038 std::int32_t Term::Private::Sigwinch::get() noexcept
00039 {
00040 #if defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00041   return Term::Private::m_signalStatus;
00042 #else
00043   return m_fd;
00044 #endif
00045 }
00046
00047 std::int32_t Term::Private::Sigwinch::m_fd{-1};
00048
00049 void Term::Private::Sigwinch::registerSigwinch()
00050 {
00051 #if defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00052   struct sigaction sa;
00053   Term::Private::Errno().check_if(sigemptyset(&sa.sa_mask) !=
      0).throw_exception("sigemptyset(&sa.sa_mask)");
00054   sa.sa_flags   = {0};
00055   sa.sa_handler = {Term::Private::sigwinchHandler};
00056   Term::Private::Errno().check_if(sigaction(SIGWINCH, &sa, nullptr) !=
      0).throw_exception("sigaction(SIGWINCH, &sa, nullptr)");
00057 #elif defined(__linux__)
00058   ::sigset_t windows_event = {};
00059   Term::Private::Errno().check_if(sigemptyset(&windows_event) !=
      0).throw_exception("sigemptyset(&windows_event)");
00060   Term::Private::Errno().check_if(sigaddset(&windows_event, SIGWINCH) !=
      0).throw_exception("sigaddset(&windows_event, SIGWINCH)");
00061   Term::Private::Errno().check_if((m_fd = ::signalfd(-1, &windows_event, SFD_NONBLOCK | SFD_CLOEXEC))
      == -1).throw_exception("m_fd = ::signalfd(-1, &windows_event, SFD_NONBLOCK | SFD_CLOEXEC)");
00062 #endif
00063 }
00064
00065 void Term::Private::Sigwinch::blockSigwinch()
00066 {
00067 #if !defined(_WIN32)
00068   ::sigset_t windows_event = {};
00069   Term::Private::Errno().check_if(sigemptyset(&windows_event) !=
      0).throw_exception("sigemptyset(&windows_event)");
00070   Term::Private::Errno().check_if(sigaddset(&windows_event, SIGWINCH) !=
      0).throw_exception("sigaddset(&windows_event, SIGWINCH)");
00071   Term::Private::Errno().check_if(::pthread_sigmask(SIG_BLOCK, &windows_event, nullptr) !=
      0).throw_exception("::pthread_sigmask(SIG_BLOCK, &windows_event, nullptr)");
00072 #endif
00073 }
00074
00075 void Term::Private::Sigwinch::unblockSigwinch()
```

```
00076 {
00077 #if !defined(_WIN32)
00078   ::sigset_t windows_event = {};
00079   Term::Private::Errno().check_if(sigemptyset(&windows_event) !=
      0).throw_exception("sigemptyset(&windows_event)");
00080   Term::Private::Errno().check_if(sigaddset(&windows_event, SIGWINCH) !=
      0).throw_exception("sigaddset(&windows_event, SIGWINCH)");
00081   Term::Private::Errno().check_if(::pthread_sigmask(SIG_UNBLOCK, &windows_event, nullptr) !=
      0).throw_exception("::pthread_sigmask(SIG_UNBLOCK, &windows_event, nullptr)");
00082 #endif
00083 }
00084
00085 bool Term::Private::Sigwinch::isSigwinch(const std::int32_t& file_descriptor) noexcept
00086 {
00087 #if defined(__APPLE__) || defined(__wasm__) || defined(__wasm) || defined(__EMSCRIPTEN__)
00088   if(Term::Private::m_signalStatus == 1)
00089   {
00090     static_cast<void>(file_descriptor);  // suppress warning
00091     Term::Private::m_signalStatus = {0};
00092     return true;
00093   }
00094   return false;
00095 #elif defined(__linux__)
00096   if(m_fd == file_descriptor)
00097   {
00098     // read it to clean
00099     ::signalfd_siginfo fdsi = {};
00100     ::read(m_fd, &fdsi, sizeof(fdsi));
00101     return true;
00102   }
00103   return false;
00104 #else
00105   static_cast<void>(file_descriptor);  // suppress warning
00106   return false;
00107 #endif
00108 }
```

9.95 cpp-terminal/private/sigwinch.hpp File Reference

`#include <cstdint>`

Classes

- class Term::Private::Sigwinch

Namespaces

- namespace Term
- namespace Term::Private

9.96 sigwinch.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstdint>
00013
00014 namespace Term
00015 {
```

```
00016 namespace Private
00017 {
00018
00019 class Sigwinch
00020 {
00021 public:
00022   static void         registerSigwinch();
00023   static void         blockSigwinch();
00024   static void         unblockSigwinch();
00025   static bool         isSigwinch(const std::int32_t& file_descriptor = -1) noexcept;
00026   static std::int32_t get() noexcept;
00027
00028 private:
00029   static std::int32_t m_fd;
00030 };
00031
00032 }  // namespace Private
00033 }  // namespace Term
```

9.97 cpp-terminal/private/terminal_impl.cpp File Reference

```
#include "cpp-terminal/cursor.hpp"
#include "cpp-terminal/private/env.hpp"
#include "cpp-terminal/private/exception.hpp"
#include "cpp-terminal/private/file.hpp"
#include "cpp-terminal/terminal.hpp"
#include <io.h>
#include <windows.h>
```

Macros

- #define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
- #define DISABLE_NEWLINE_AUTO_RETURN 0x0008
- #define ENABLE_VIRTUAL_TERMINAL_INPUT 0x0200

9.97.1 Macro Definition Documentation

DISABLE_NEWLINE_AUTO_RETURN

```
#define DISABLE_NEWLINE_AUTO_RETURN 0x0008
```

Definition at line 23 of file terminal_impl.cpp.

ENABLE_VIRTUAL_TERMINAL_INPUT

```
#define ENABLE_VIRTUAL_TERMINAL_INPUT 0x0200
```

Definition at line 26 of file terminal_impl.cpp.

ENABLE_VIRTUAL_TERMINAL_PROCESSING

```
#define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
```

Definition at line 20 of file terminal_impl.cpp.

9.98 terminal_impl.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/cursor.hpp"
00011 #include "cpp-terminal/private/env.hpp"
00012 #include "cpp-terminal/private/exception.hpp"
00013 #include "cpp-terminal/private/file.hpp"
00014 #include "cpp-terminal/terminal.hpp"
00015
00016 #if defined(_WIN32)
00017   #include <io.h>
00018   #include <windows.h>
00019   #if !defined(ENABLE_VIRTUAL_TERMINAL_PROCESSING)
00020     #define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
00021   #endif
00022   #if !defined(DISABLE_NEWLINE_AUTO_RETURN)
00023     #define DISABLE_NEWLINE_AUTO_RETURN 0x0008
00024   #endif
00025   #if !defined(ENABLE_VIRTUAL_TERMINAL_INPUT)
00026     #define ENABLE_VIRTUAL_TERMINAL_INPUT 0x0200
00027   #endif
00028 #else
00029   #include <termios.h>
00030 #endif
00031
00032 void Term::Terminal::set_unset_utf8()
00033 {
00034   static bool enabled{false};
00035 #if defined(_WIN32)
00036   static UINT out_code_page{0};
00037   static UINT in_code_page{0};
00038   if(!enabled)
00039   {
00040     if((out_code_page = GetConsoleOutputCP()) == 0) throw
      Term::Private::WindowsException(GetLastError(), "GetConsoleOutputCP()");
00041     if(!SetConsoleOutputCP(CP_UTF8)) throw Term::Private::WindowsException(GetLastError(),
      "SetConsoleOutputCP(CP_UTF8)");
00042     if((in_code_page = GetConsoleCP()) == 0) throw Term::Private::WindowsException(GetLastError(),
      "GetConsoleCP()");
00043     if(!SetConsoleCP(CP_UTF8)) throw Term::Private::WindowsException(GetLastError(),
      "SetConsoleCP(CP_UTF8)");
00044     enabled = true;
00045   }
00046   else
00047   {
00048     if(!SetConsoleOutputCP(out_code_page)) throw Term::Private::WindowsException(GetLastError(),
      "SetConsoleOutputCP(out_code_page)");
00049     if(!SetConsoleCP(in_code_page)) throw Term::Private::WindowsException(GetLastError(),
      "SetConsoleCP(in_code_page)");
00050   }
00051 #else
00052   if(!enabled)
00053   {
00054     const Term::Cursor cursor_before{Term::cursor_position()};
00055     Term::Private::out.write("\u001b%G");  // Try to activate UTF-8 (NOT warranty)
00056     const std::string  read{Term::Private::in.read()};
00057     const Term::Cursor cursor_after{Term::cursor_position()};
00058     const std::size_t  moved{cursor_after.column() - cursor_before.column()};
00059     std::string        rem;
00060     rem.reserve(moved * 3);
00061     for(std::size_t i = 0; i != moved; ++i) { rem += "\b \b"; }
00062     Term::Private::out.write(rem);
00063     enabled = 0 == moved;
00064   }
00065   else
00066   {
00067     // Does not return the original charset but, the default defined by standard ISO 8859-1 (ISO
      2022);
00068     Term::Private::out.write("\u001b%@");
00069   }
00070 #endif
00071 }
00072
00073 void Term::Terminal::store_and_restore() noexcept
00074 try
00075 {
00076   static bool enabled{false};
```

```
00077 #if defined(_WIN32)
00078   static DWORD originalOut{0};
00079   static DWORD originalIn{0};
00080   if(!enabled)
00081   {
00082     Term::Private::WindowsError().check_if(GetConsoleMode(Private::out.handle(), &originalOut) ==
      0).throw_exception("GetConsoleMode(Private::out.handle(), &originalOut)");
00083     Term::Private::WindowsError().check_if(GetConsoleMode(Private::in.handle(), &originalIn) ==
      0).throw_exception("GetConsoleMode(Private::in.handle(), &originalIn)");
00084     DWORD in{static_cast<DWORD>((originalIn & ~(ENABLE_QUICK_EDIT_MODE | setFocusEvents() |
      setMouseEvents())) | (ENABLE_EXTENDED_FLAGS))};
00085     DWORD out{originalOut};
00086     // Check if ENABLE_VIRTUAL_TERMINAL_PROCESSING | DISABLE_NEWLINE_AUTO_RETURN can be activated, if
      not we are a legacy terminal.
00087     DWORD test = out;
00088     test |= ENABLE_VIRTUAL_TERMINAL_PROCESSING;
00089     if(!SetConsoleMode(Private::out.handle(), test)) { SetConsoleMode(Private::out.handle(), out); }
00090     else
00091     {
00092       out |= ENABLE_VIRTUAL_TERMINAL_PROCESSING | DISABLE_NEWLINE_AUTO_RETURN;
00093       in |= ENABLE_VIRTUAL_TERMINAL_INPUT;
00094     }
00095     if(!SetConsoleMode(Private::out.handle(), out)) { throw
      Term::Private::WindowsException(GetLastError(), "SetConsoleMode(Private::out.handle()"); }
00096     if(!SetConsoleMode(Private::in.handle(), in)) { throw
      Term::Private::WindowsException(GetLastError(), "SetConsoleMode(Private::in.handle(), in)"); }
00097     enabled = true;
00098   }
00099   else
00100   {
00101     if(!SetConsoleMode(Private::out.handle(), originalOut)) { throw
      Term::Private::WindowsException(GetLastError(), "SetConsoleMode(Private::out.handle(), originalOut)");
      }
00102     if(!SetConsoleMode(Private::in.handle(), originalIn)) { throw
      Term::Private::WindowsException(GetLastError(), "SetConsoleMode(Private::in.handle(), originalIn)"); }
00103   }
00104 #else
00105   static termios orig_termios;
00106   if(!enabled)
00107   {
00108     if(!Private::out.null()) { Term::Private::Errno().check_if(tcgetattr(Private::out.fd(),
      &orig_termios) == -1).throw_exception("tcgetattr() failed"); }
00109     enabled = true;
00110   }
00111   else
00112   {
00113     unsetMouseEvents();
00114     unsetFocusEvents();
00115     if(!Private::out.null()) { Term::Private::Errno().check_if(tcsetattr(Private::out.fd(), TCSAFLUSH,
      &orig_termios) == -1).throw_exception("tcsetattr() failed in destructor"); }
00116   }
00117 #endif
00118 }
00119 catch(...)
00120 {
00121   ExceptionHandler(Private::ExceptionDestination::StdErr);
00122 }
00123
00124 std::size_t Term::Terminal::setMouseEvents()
00125 {
00126 #if defined(_WIN32)
00127   return static_cast<std::size_t>(ENABLE_MOUSE_INPUT);
00128 #else
00129   return Term::Private::out.write("\u001b[?1002h\u001b[?1003h\u001b[?1006h");
00130 #endif
00131 }
00132
00133 std::size_t Term::Terminal::unsetMouseEvents()
00134 {
00135 #if defined(_WIN32)
00136   return static_cast<std::size_t>(ENABLE_MOUSE_INPUT);
00137 #else
00138   return Term::Private::out.write("\u001b[?1006l\u001b[?1003l\u001b[?1002l");
00139 #endif
00140 }
00141
00142 std::size_t Term::Terminal::setFocusEvents()
00143 {
00144 #if defined(_WIN32)
00145   return static_cast<std::size_t>(ENABLE_WINDOW_INPUT);
00146 #else
00147   return Term::Private::out.write("\u001b[?1004h");
00148 #endif
00149 }
00150
00151 std::size_t Term::Terminal::unsetFocusEvents()
00152 {
```

```
00153 #if defined(_WIN32)
00154   return static_cast<std::size_t>(ENABLE_WINDOW_INPUT);
00155 #else
00156   return Term::Private::out.write("\u001b[?1004l");
00157 #endif
00158 }
00159
00160 void Term::Terminal::setMode() const
00161 {
00162   static bool activated{false};
00163 #if defined(_WIN32)
00164   static DWORD flags{0};
00165   if(!activated)
00166   {
00167     if(!Private::out.null())
00168       if(!GetConsoleMode(Private::in.handle(), &flags)) { throw
      Term::Private::WindowsException(GetLastError()); }
00169     activated = true;
00170     return;
00171   }
00172   DWORD send = flags;
00173   if(m_options.has(Option::Raw))
00174   {
00175     send &= ~(ENABLE_LINE_INPUT | ENABLE_ECHO_INPUT | ENABLE_PROCESSED_INPUT);
00176     send |= (setFocusEvents() | setMouseEvents());
00177   }
00178   else if(m_options.has(Option::Cooked))
00179   {
00180     send |= (ENABLE_LINE_INPUT | ENABLE_ECHO_INPUT | ENABLE_PROCESSED_INPUT);
00181     send &= ~(setFocusEvents() | setMouseEvents());
00182   }
00183   if(m_options.has(Option::NoSignalKeys)) { send &= ~ENABLE_PROCESSED_INPUT; }
00184   else if(m_options.has(Option::SignalKeys)) { send |= ENABLE_PROCESSED_INPUT; }
00185   if(!Private::out.null())
00186     if(!SetConsoleMode(Private::in.handle(), send)) { throw
      Term::Private::WindowsException(GetLastError()); }
00187 #else
00188   if(!Private::out.null())
00189   {
00190     static ::termios raw = {};
00191     if(!activated)
00192     {
00193       Term::Private::Errno().check_if(tcgetattr(Private::out.fd(), &raw) ==
      -1).throw_exception("tcgetattr(Private::out.fd(), &raw)");
00194       raw.c_cflag &= ~static_cast<std::size_t>(CSIZE | PARENB);
00195       raw.c_cflag |= CS8;
00196       raw.c_cc[VMIN]  = 1;
00197       raw.c_cc[VTIME] = 0;
00198       activated       = true;
00199       return;
00200     }
00201     ::termios send = raw;
00202     if(m_options.has(Option::Raw))
00203     {
00204       send.c_iflag &= ~static_cast<std::size_t>(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR |
      ICRNL | IXON | INPCK);
00205       // This disables output post-processing, requiring explicit \r\n. We
00206       // keep it enabled, so that in C++, one can still just use std::endl
00207       // for EOL instead of "\r\n".
00208       //send.c_oflag &= ~static_cast<std::size_t>(OPOST);
00209       send.c_lflag &= ~static_cast<std::size_t>(ECHO | ECHONL | ICANON | ISIG | IEXTEN);
00210       setMouseEvents();
00211       setFocusEvents();
00212     }
00213     else if(m_options.has(Option::Cooked))
00214     {
00215       send = raw;
00216       unsetMouseEvents();
00217       unsetFocusEvents();
00218     }
00219     if(m_options.has(Option::NoSignalKeys)) { send.c_lflag &= ~static_cast<std::size_t>(ISIG); }
      //FIXME need others flags !
00220     else if(m_options.has(Option::NoSignalKeys)) { send.c_lflag |= ISIG; }
00221     Term::Private::Errno().check_if(tcsetattr(Private::out.fd(), TCSAFLUSH, &send) ==
      -1).throw_exception("tcsetattr(Private::out.fd(), TCSAFLUSH, &send)");
00222   }
00223 #endif
00224 }
```

9.99 cpp-terminal/terminal_impl.cpp File Reference

```
#include "cpp-terminal/terminal_impl.hpp"
#include "cpp-terminal/cursor.hpp"
```

```
#include "cpp-terminal/options.hpp"
#include "cpp-terminal/private/exception.hpp"
#include "cpp-terminal/private/file.hpp"
#include "cpp-terminal/private/sigwinch.hpp"
#include "cpp-terminal/screen.hpp"
#include "cpp-terminal/style.hpp"
#include "cpp-terminal/terminal.hpp"
```

9.100 terminal_impl.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/terminal_impl.hpp"
00011
00012 #include "cpp-terminal/cursor.hpp"
00013 #include "cpp-terminal/options.hpp"
00014 #include "cpp-terminal/private/exception.hpp"
00015 #include "cpp-terminal/private/file.hpp"
00016 #include "cpp-terminal/private/sigwinch.hpp"
00017 #include "cpp-terminal/screen.hpp"
00018 #include "cpp-terminal/style.hpp"
00019 #include "cpp-terminal/terminal.hpp"  //FIXME avoid recursion
00020
00021 Term::Options Term::Terminal::getOptions() const noexcept { return m_options; }
00022
00023 Term::Terminal::Terminal() noexcept
00024 try
00025 {
00026   Term::Private::Sigwinch::blockSigwinch();
00027   Term::Private::Sigwinch::registerSigwinch();
00028   store_and_restore();
00029   setMode();  //Save the default cpp-terminal mode done in store_and_restore();
00030   set_unset_utf8();
00031 }
00032 catch(...)
00033 {
00034   ExceptionHandler(Private::ExceptionDestination::StdErr);
00035 }
00036
00037 Term::Terminal::~Terminal() noexcept
00038 try
00039 {
00040   if(getOptions().has(Option::ClearScreen)) { Term::Private::out.write(clear_buffer() +
      style(Style::Reset) + cursor_move(1, 1) + screen_load()); }
00041   if(getOptions().has(Option::NoCursor)) { Term::Private::out.write(cursor_on()); }
00042   set_unset_utf8();
00043   store_and_restore();
00044   unsetFocusEvents();
00045   unsetMouseEvents();
00046 }
00047 catch(...)
00048 {
00049   ExceptionHandler(Private::ExceptionDestination::StdErr);
00050 }
00051
00052 void Term::Terminal::applyOptions() const
00053 {
00054   if(getOptions().has(Option::ClearScreen)) { Term::Private::out.write(screen_save() + clear_buffer()
      + style(Style::Reset) + cursor_move(1, 1)); }
00055   if(getOptions().has(Option::NoCursor)) { Term::Private::out.write(cursor_off()); }
00056   setMode();
00057 }
```

9.101 cpp-terminal/private/terminfo.cpp File Reference

```
#include <windows.h>
#include "cpp-terminal/cursor.hpp"
```

```
#include "cpp-terminal/private/env.hpp"
#include "cpp-terminal/private/file.hpp"
#include "cpp-terminal/terminfo.hpp"
#include <cstddef>
#include <string>
```

Macros

- #define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004

Functions

- bool WindowsVersionGreater (const DWORD &major, const DWORD &minor, const DWORD &patch)

9.101.1 Macro Definition Documentation

ENABLE_VIRTUAL_TERMINAL_PROCESSING

```
#define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
```

9.101.2 Function Documentation

WindowsVersionGreater()

```
bool WindowsVersionGreater (
            const DWORD & major,
            const DWORD & minor,
            const DWORD & patch )
```

Definition at line 50 of file terminfo.cpp.

```
00051 {
00052   #if defined(_MSC_VER)
00053     #pragma warning(push)
00054     #pragma warning(disable : 4191)
00055   #else
00056     #pragma GCC diagnostic push
00057     #pragma GCC diagnostic ignored "-Wcast-function-type"
00058   #endif
00059   NTSTATUS(WINAPI * getVersion)
00060   (PRTL_OSVERSIONINFOW) =
      (reinterpret_cast<NTSTATUS(WINAPI*)(PRTL_OSVERSIONINFOW)>(GetProcAddress(GetModuleHandle(TEXT("ntdll.dll")),
      "RtlGetVersion")));
00061   #if defined(_MSC_VER)
00062     #pragma warning(pop)
00063   #else
00064     #pragma GCC diagnostic pop
00065   #endif
00066   if(getVersion != nullptr)
00067   {
00068     RTL_OSVERSIONINFOW rovi;
00069     rovi.dwOSVersionInfoSize = sizeof(rovi);
00070     if(getVersion(&rovi) == 0)
00071     {
00072       if(rovi.dwMajorVersion > major || (rovi.dwMajorVersion == major && (rovi.dwMinorVersion > minor
      || (rovi.dwMinorVersion == minor && rovi.dwBuildNumber >= patch)))) return true;
00073       else
00074         return false;
00075     }
00076   }
00077   return false;
00078 }
```

9.102 terminfo.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #ifdef _WIN32
00011   #include <windows.h>
00012 #endif
00013
00014 #include "cpp-terminal/cursor.hpp"
00015 #include "cpp-terminal/private/env.hpp"
00016 #include "cpp-terminal/private/file.hpp"
00017 #include "cpp-terminal/terminfo.hpp"
00018
00019 #include <cstddef>
00020 #include <string>
00021
00022 Term::Terminfo::ColorMode Term::Terminfo::m_colorMode{ColorMode::Unset};
00023 Term::Terminfo::Booleans  Term::Terminfo::m_booleans{};
00024 Term::Terminfo::Integers  Term::Terminfo::m_integers{};
00025 Term::Terminfo::Strings   Term::Terminfo::m_strings{};
00026
00027 bool Term::Terminfo::get(const Term::Terminfo::Bool& key)
00028 {
00029   check();
00030   return m_booleans[static_cast<std::size_t>(key)];
00031 }
00032
00033 std::uint32_t Term::Terminfo::get(const Term::Terminfo::Integer& key)
00034 {
00035   check();
00036   return m_integers[static_cast<std::size_t>(key)];
00037 }
00038
00039 std::string Term::Terminfo::get(const Term::Terminfo::String& key)
00040 {
00041   check();
00042   return m_strings[static_cast<std::size_t>(key)];
00043 }
00044
00045 void Term::Terminfo::set(const Term::Terminfo::Bool& key, const bool& value) {
      m_booleans[static_cast<std::size_t>(key)] = value; }
00046 void Term::Terminfo::set(const Term::Terminfo::Integer& key, const std::uint32_t& value) {
      m_integers[static_cast<std::size_t>(key)] = value; }
00047 void Term::Terminfo::set(const Term::Terminfo::String& key, const std::string& value) {
      m_strings[static_cast<std::size_t>(key)] = value; }
00048
00049 #if defined(_WIN32)
00050 bool WindowsVersionGreater(const DWORD& major, const DWORD& minor, const DWORD& patch)
00051 {
00052   #if defined(_MSC_VER)
00053     #pragma warning(push)
00054     #pragma warning(disable : 4191)
00055   #else
00056     #pragma GCC diagnostic push
00057     #pragma GCC diagnostic ignored "-Wcast-function-type"
00058   #endif
00059   NTSTATUS(WINAPI * getVersion)
00060   (PRTL_OSVERSIONINFOW) =
      (reinterpret_cast<NTSTATUS(WINAPI*)(PRTL_OSVERSIONINFOW)>(GetProcAddress(GetModuleHandle(TEXT("ntdll.dll")),
      "RtlGetVersion")));
00061   #if defined(_MSC_VER)
00062     #pragma warning(pop)
00063   #else
00064     #pragma GCC diagnostic pop
00065   #endif
00066   if(getVersion != nullptr)
00067   {
00068     RTL_OSVERSIONINFOW rovi;
00069     rovi.dwOSVersionInfoSize = sizeof(rovi);
00070     if(getVersion(&rovi) == 0)
00071     {
00072       if(rovi.dwMajorVersion > major || (rovi.dwMajorVersion == major && (rovi.dwMinorVersion > minor
      || (rovi.dwMinorVersion == minor && rovi.dwBuildNumber >= patch)))) return true;
00073       else
00074         return false;
00075     }
00076   }
00077   return false;
```

```
00078 }
00079 #endif
00080
00081 void Term::Terminfo::checkLegacy()
00082 {
00083 #if defined(_WIN32)
00084   #ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING
00085     #define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
00086   #endif
00087   if(!m_booleans[static_cast<std::size_t>(Terminfo::Bool::ControlSequences)]) {
      set(Terminfo::Bool::Legacy, true); }
00088   else
00089   {
00090     DWORD dwOriginalOutMode{0};
00091     GetConsoleMode(Private::out.handle(), &dwOriginalOutMode);
00092     if(!SetConsoleMode(Private::out.handle(), dwOriginalOutMode | ENABLE_VIRTUAL_TERMINAL_PROCESSING))
      { set(Terminfo::Bool::Legacy, true); }
00093     else
00094     {
00095       SetConsoleMode(Private::out.handle(), dwOriginalOutMode);
00096       set(Terminfo::Bool::Legacy, false);
00097     }
00098   }
00099 #else
00100   set(Terminfo::Bool::Legacy, false);
00101 #endif
00102 }
00103
00104 void Term::Terminfo::checkTermEnv() { set(Terminfo::String::TermEnv, Private::getenv("TERM").second);
      }
00105
00106 void Term::Terminfo::checkTerminalName()
00107 {
00108   std::string name;
00109   name = Private::getenv("TERM_PROGRAM").second;
00110   if(name.empty()) { name = Private::getenv("TERMINAL_EMULATOR").second; }
00111   if(Private::getenv("ANSICON").first) { name = "ansicon"; }
00112   set(Term::Terminfo::String::TermName, name);
00113 }
00114
00115 void Term::Terminfo::checkTerminalVersion() { set(Terminfo::String::TermVersion,
      Private::getenv("TERM_PROGRAM_VERSION").second); }
00116
00117 void Term::Terminfo::check()
00118 {
00119   static bool checked{false};
00120   if(!checked)
00121   {
00122     checkTermEnv();
00123     checkTerminalName();
00124     checkTerminalVersion();
00125     checkControlSequences();
00126     checkLegacy();
00127     checkColorMode();
00128     checkUTF8();
00129     checked = true;
00130   }
00131 }
00132
00133 Term::Terminfo::ColorMode Term::Terminfo::getColorMode()
00134 {
00135   checkColorMode();
00136   return m_colorMode;
00137 }
00138
00139 Term::Terminfo::Terminfo() { check(); }
00140
00141 void Term::Terminfo::checkColorMode()
00142 {
00143   std::string name{m_strings[static_cast<std::size_t>(Terminfo::String::TermName)]};
00144   if(name == "Apple_Terminal") { m_colorMode = Term::Terminfo::ColorMode::Bit8; }
00145   else if(name == "JetBrains-JediTerm") { m_colorMode = Term::Terminfo::ColorMode::Bit24; }
00146   else if(name == "vscode") { m_colorMode = Term::Terminfo::ColorMode::Bit24; }
00147   else if(name == "linux") { m_colorMode = Term::Terminfo::ColorMode::Bit4; }
00148   else if(name == "ansicon") { m_colorMode = Term::Terminfo::ColorMode::Bit4; }
00149   else if(m_strings[static_cast<std::size_t>(Terminfo::String::TermEnv)] == "linux") { m_colorMode =
      Term::Terminfo::ColorMode::Bit4; }
00150 #if defined(_WIN32)
00151   else if(WindowsVersionGreater(10, 0, 10586) &&
      !m_booleans[static_cast<std::size_t>(Terminfo::Bool::Legacy)]) { m_colorMode =
      Term::Terminfo::ColorMode::Bit24; }
00152   else if(m_booleans[static_cast<std::size_t>(Terminfo::Bool::Legacy)]) { m_colorMode =
      Term::Terminfo::ColorMode::Bit4; }
00153 #endif
00154   else { m_colorMode = Term::Terminfo::ColorMode::Bit24; }
00155   std::string colorterm = Private::getenv("COLORTERM").second;
00156   if((colorterm == "truecolor" || colorterm == "24bit") && m_colorMode != ColorMode::Unset) {
```

```
      m_colorMode = Term::Terminfo::ColorMode::Bit24; }
00157 }
00158
00159 void Term::Terminfo::checkControlSequences()
00160 {
00161 #ifdef _WIN32
00162   if(WindowsVersionGreater(10, 0, 10586)) { set(Term::Terminfo::Bool::ControlSequences, true); }
00163   else { set(Term::Terminfo::Bool::ControlSequences, false); }
00164 #else
00165   set(Term::Terminfo::Bool::ControlSequences, true);
00166 #endif
00167 }
00168
00169 void Term::Terminfo::checkUTF8()
00170 {
00171 #if defined(_WIN32)
00172   (GetConsoleOutputCP() == CP_UTF8 && GetConsoleCP() == CP_UTF8) ? set(Terminfo::Bool::UTF8, true) :
      set(Terminfo::Bool::UTF8, false);
00173 #else
00174   Term::Cursor cursor_before{Term::cursor_position()};
00175   Term::Private::out.write("\xe2\x82\xac");  // € 3bits in utf8 one character
00176   std::string  read{Term::Private::in.read()};
00177   Term::Cursor cursor_after{Term::cursor_position()};
00178   std::size_t  moved{cursor_after.column() - cursor_before.column()};
00179   if(moved == 1) { set(Terminfo::Bool::UTF8, true); }
00180   else { set(Terminfo::Bool::UTF8, false); }
00181   for(std::size_t i = 0; i != moved; ++i) { Term::Private::out.write("\b \b"); }
00182 #endif
00183 }
```

9.103 cpp-terminal/private/tty.cpp File Reference

```
#include <cstdio>
#include <io.h>
#include "cpp-terminal/tty.hpp"
```

9.104 tty.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include <cstdio>
00011
00012 #ifdef _WIN32
00013   #include <io.h>
00014 #else
00015   #include <unistd.h>
00016 #endif
00017
00018 #include "cpp-terminal/tty.hpp"
00019
00020 namespace
00021 {
00022 bool is_a_tty(const FILE* fd)
00023 {
00024 #ifdef _WIN32
00025   return static_cast<bool>(_isatty(_fileno(const_cast<FILE*>(fd))));
00026 #else
00027   return ::isatty(::fileno(const_cast<FILE*>(fd)));
00028 #endif
00029 }
00030 }  // namespace
00031
00032 bool Term::is_stdin_a_tty() { return ::is_a_tty(stdin); }
00033
00034 bool Term::is_stdout_a_tty() { return ::is_a_tty(stdout); }
00035
00036 bool Term::is_stderr_a_tty() { return ::is_a_tty(stderr); }
```

9.105 cpp-terminal/private/unicode.cpp File Reference

```
#include "cpp-terminal/private/unicode.hpp"
#include "cpp-terminal/private/exception.hpp"
#include <limits>
#include <windows.h>
#include <array>
```

9.106 unicode.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/private/unicode.hpp"
00011
00012 #include "cpp-terminal/private/exception.hpp"
00013
00014 #if defined(_WIN32)
00015   #include <limits>
00016   #include <windows.h>
00017 #endif
00018
00019 #include <array>
00020
00021 #if defined(_WIN32)
00022 std::string Term::Private::to_narrow(const std::wstring& in)
00023 {
00024   if(in.empty()) return std::string();
00025   static constexpr DWORD flag{WC_ERR_INVALID_CHARS};
00026   std::size_t            in_size{in.size()};
00027   if(in_size > static_cast<size_t>((std::numeric_limits<int>::max)())) throw Term::Exception("String
      size is to big " + std::to_string(in_size) + "/" + std::to_string((std::numeric_limits<int>::max)()));
00028   const int ret_size{::WideCharToMultiByte(CP_UTF8, flag, in.data(), static_cast<int>(in_size),
      nullptr, 0, nullptr, nullptr)};
00029   if(ret_size == 0) throw Term::Private::WindowsException(::GetLastError());
00030   std::string ret(static_cast<std::size_t>(ret_size), '\0');
00031   int         ret_error{::WideCharToMultiByte(CP_UTF8, flag, in.data(), static_cast<int>(in_size),
      &ret[0], ret_size, nullptr, nullptr)};
00032   if(ret_error == 0) throw Term::Private::WindowsException(::GetLastError());
00033   return ret;
00034 }
00035
00036 std::wstring Term::Private::to_wide(const std::string& in)
00037 {
00038   if(in.empty()) return std::wstring();
00039   static constexpr DWORD flag{MB_ERR_INVALID_CHARS};
00040   std::size_t            in_size{in.size()};
00041   if(in_size > static_cast<size_t>((std::numeric_limits<int>::max)())) throw Term::Exception("String
      size is to big " + std::to_string(in_size) + "/" + std::to_string((std::numeric_limits<int>::max)()));
00042   const int ret_size{::MultiByteToWideChar(CP_UTF8, flag, in.data(), static_cast<int>(in_size),
      nullptr, 0)};
00043   if(ret_size == 0) throw Term::Private::WindowsException(::GetLastError());
00044   std::wstring ret(static_cast<std::size_t>(ret_size), '\0');
00045   int          ret_error{::MultiByteToWideChar(CP_UTF8, flag, in.data(), static_cast<int>(in_size),
      &ret[0], ret_size)};
00046   if(ret_error == 0) throw Term::Private::WindowsException(::GetLastError());
00047   return ret;
00048 }
00049 #endif
00050
00051 std::string Term::Private::utf32_to_utf8(const char32_t& codepoint, const bool& exception)
00052 {
00053   static const constexpr std::array<std::uint32_t, 4> size{0x7F, 0x07FF, 0xFFFF, 0x10FFFF};
00054   static const constexpr std::uint8_t                  mask{0x80};
00055   static const constexpr std::uint8_t                  add{0x3F};
00056   static const constexpr std::array<std::uint8_t, 3>   mask_first{0x1F, 0x0F, 0x07};
00057   static const constexpr std::array<std::uint8_t, 3>   add_first{0xC0, 0xE0, 0xF0};
00058   static const constexpr std::array<std::uint8_t, 4>   shift{0, 6, 12, 18};
00059   static const constexpr std::uint8_t                  max_size{4};
00060   std::string                                          ret;
```

```
00061   ret.reserve(max_size);
00062   if(codepoint <= size[0]) { ret = {static_cast<char>(codepoint)}; }  // Plain ASCII
00063   else if(codepoint <= size[1]) { ret = {static_cast<char>(((codepoint » shift[1]) & mask_first[0]) |
      add_first[0]), static_cast<char>(((codepoint » shift[0]) & add) | mask)}; }
00064   else if(codepoint <= size[2]) { ret = {static_cast<char>(((codepoint » shift[2]) & mask_first[1]) |
      add_first[1]), static_cast<char>(((codepoint » shift[1]) & add) | mask), static_cast<char>(((codepoint
      » shift[0]) & add) | mask)}; }
00065   else if(codepoint <= size[3]) { ret = {static_cast<char>(((codepoint » shift[3]) & mask_first[2]) |
      add_first[2]), static_cast<char>(((codepoint » shift[2]) & add) | mask), static_cast<char>(((codepoint
      » shift[1]) & add) | mask), static_cast<char>(((codepoint » shift[0]) & add) | mask)}; }
00066   else if(exception) { throw Term::Exception("Invalid UTF32 codepoint."); }
00067   else { ret = "\xEF\xBF\xBD"; }
00068   return ret;
00069 }
00070
00071 std::string Term::Private::utf32_to_utf8(const std::u32string& str, const bool& exception)
00072 {
00073   std::string ret;
00074   for(const char32_t codepoint: str) { ret.append(utf32_to_utf8(codepoint, exception)); }
00075   return ret;
00076 }
```

9.107 cpp-terminal/private/unicode.hpp File Reference

`#include <string>`

Namespaces

- namespace Term
- namespace Term::Private

Functions

- std::string Term::Private::to_narrow (const std::wstring &wstr)
- std::wstring Term::Private::to_wide (const std::string &str)
- std::string Term::Private::utf32_to_utf8 (const char32_t &codepoint, const bool &exception=false)

  Encode a codepoint using UTF-8 ***std::string*** .
- std::string Term::Private::utf32_to_utf8 (const std::u32string &str, const bool &exception=false)

  Encode a ***std::u32string*** *into UTF-8* ***std::string*** .

9.108 unicode.hpp

Go to the documentation of this file.

```
00001
00002 /*
00003 * cpp-terminal
00004 * C++ library for writing multi-platform terminal applications.
00005 *
00006 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00007 *
00008 * SPDX-License-Identifier: MIT
00009 */
00010
00011 #pragma once
00012
00013 #include <string>
00014
00015 namespace Term
00016 {
00017 namespace Private
00018 {
00019
00020 // utf16 is useless and wstring too so utf16 inside wstring is useless^2 but windows use it so define
      this functions to deal with it and less the user forget it.
```

```
00021 #if defined(_WIN32)
00022 std::string  to_narrow(const std::wstring& wstr);
00023 std::wstring to_wide(const std::string& str);
00024 #endif
00025
00034 std::string utf32_to_utf8(const char32_t& codepoint, const bool& exception = false);
00035
00044 std::string utf32_to_utf8(const std::u32string& str, const bool& exception = false);
00045
00046 }  // namespace Private
00047 }  // namespace Term
```

9.109 cpp-terminal/prompt.cpp File Reference

```
#include "cpp-terminal/prompt.hpp"
#include "cpp-terminal/cursor.hpp"
#include "cpp-terminal/event.hpp"
#include "cpp-terminal/exception.hpp"
#include "cpp-terminal/input.hpp"
#include "cpp-terminal/iostream.hpp"
#include "cpp-terminal/key.hpp"
#include "cpp-terminal/private/conversion.hpp"
#include "cpp-terminal/private/macros.hpp"
#include "cpp-terminal/screen.hpp"
#include "cpp-terminal/terminal.hpp"
#include "cpp-terminal/tty.hpp"
#include <iostream>
```

Functions

- char32_t UU (const std::string &s)

9.109.1 Function Documentation

UU()

```
char32_t UU (
            const std::string & s )
```

Definition at line 168 of file prompt.cpp.

```
00169 {
00170   std::u32string s2 = Term::Private::utf8_to_utf32(s);
00171   if(s2.size() != 1) throw Term::Exception("U(s): s not a codepoint.");
00172   return s2[0];
00173 }
```

9.110 prompt.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
```

```
00010 #include "cpp-terminal/prompt.hpp"
00011
00012 #include "cpp-terminal/cursor.hpp"
00013 #include "cpp-terminal/event.hpp"
00014 #include "cpp-terminal/exception.hpp"
00015 #include "cpp-terminal/input.hpp"
00016 #include "cpp-terminal/iostream.hpp"
00017 #include "cpp-terminal/key.hpp"
00018 #include "cpp-terminal/private/conversion.hpp"
00019 #include "cpp-terminal/private/macros.hpp"
00020 #include "cpp-terminal/screen.hpp"
00021 #include "cpp-terminal/terminal.hpp"
00022 #include "cpp-terminal/tty.hpp"
00023
00024 #include <iostream>
00025
00026 Term::Result Term::prompt(const std::string& message, const std::string& first_option, const
      std::string& second_option, const std::string& prompt_indicator, bool immediate)
00027 {
00028   Term::terminal.setOptions(Option::NoClearScreen, Option::NoSignalKeys, Option::Cursor,
      Term::Option::Raw);
00029   std::cout « message « " [" « first_option « '/' « second_option « ']' « prompt_indicator « ' ' «
      std::flush;
00030
00031   if(!Term::is_stdin_a_tty())
00032   {
00033     std::cout « '\n' « std::flush;
00034     return Result::Error;
00035   }
00036
00037   Term::Key key;
00038
00039   if(immediate)
00040   {
00041     while(true)
00042     {
00043       key = Term::read_event();
00044       if(key == Term::Key::NoKey) continue;
00045       if(key == Term::Key::y || key == Term::Key::Y)
00046       {
00047         std::cout « '\n' « std::flush;
00048         return Result::Yes;
00049       }
00050       else if(key == Term::Key::n || key == Term::Key::N)
00051       {
00052         std::cout « '\n' « std::flush;
00053         return Result::No;
00054       }
00055       else if(key == Term::Key::Ctrl_C || key == Term::Key::Ctrl_D)
00056       {
00057         std::cout « '\n' « std::flush;
00058         return Result::Abort;
00059       }
00060       else if(key == Term::Key::Enter)
00061       {
00062         std::cout « '\n' « std::flush;
00063         return Result::None;
00064       }
00065       else
00066       {
00067         std::cout « '\n' « std::flush;
00068         return Result::Invalid;
00069       }
00070     }
00071   }
00072   else
00073   {
00074     std::string input;
00075     while(true)
00076     {
00077       key = Term::read_event();
00078       if(key == Term::Key::NoKey) continue;
00079       if(key >= 'a' && key <= 'z')
00080       {
00081         std::cout « (char)key « std::flush;
00082         input.push_back(static_cast<char>(key));
00083       }
00084       else if(key >= 'A' && key <= 'Z')
00085       {
00086         std::cout « (char)key « std::flush;
00087         input.push_back(static_cast<char>(key.tolower()));  // convert upper case to lowercase
00088       }
00089       else if(key == Term::Key::Ctrl_C || key == Term::Key::Ctrl_D)
00090       {
00091         std::cout « '\n';
00092         return Result::Abort;
00093       }
```

```
00094       else if(key == Term::Key::Backspace)
00095       {
00096         if(input.empty() != 0)
00097         {
00098           std::cout « "\u001b[D \u001b[D" « std::flush;  // erase last line and move the cursor back
00099           input.pop_back();
00100         }
00101       }
00102       else if(key == Term::Key::Enter)
00103       {
00104         if(input == "y" || input == "yes")
00105         {
00106           std::cout « '\n' « std::flush;
00107           return Result::Yes;
00108         }
00109         else if(input == "n" || input == "no")
00110         {
00111           std::cout « '\n' « std::flush;
00112           return Result::No;
00113         }
00114         else if(input.empty())
00115         {
00116           std::cout « '\n' « std::flush;
00117           return Result::None;
00118         }
00119         else
00120         {
00121           std::cout « '\n' « std::flush;
00122           return Result::Invalid;
00123         }
00124       }
00125     }
00126   }
00127 }
00128
00129 Term::Result_simple Term::prompt_simple(const std::string& message)
00130 {
00131   switch(prompt(message, "y", "N", ":", false))
00132   {
00133     case Result::Yes: return Result_simple::Yes;
00134     case Result::Abort: return Result_simple::Abort;
00135     case Result::No:     // falls through
00136     case Result::Error:  // falls through
00137     case Result::None:   // falls through
00138     case Result::Invalid:
00139     default: return Result_simple::No;
00140   }
00141 }
00142
00143 std::string Term::concat(const std::vector<std::string>& lines)
00144 {
00145   std::string s;
00146   for(auto& line: lines) { s.append(line + "\n"); }
00147   return s;
00148 }
00149
00150 std::vector<std::string> Term::split(const std::string& s)
00151 {
00152   std::size_t              j = 0;
00153   std::vector<std::string> lines;
00154   lines.emplace_back("");
00155   if(s[s.size() - 1] != '\n') throw Term::Exception("\\n is required");
00156   for(std::size_t i = 0; i < s.size() - 1; i++)
00157   {
00158     if(s[i] == '\n')
00159     {
00160       j++;
00161       lines.emplace_back("");
00162     }
00163     else { lines[j].push_back(s[i]); }
00164   }
00165   return lines;
00166 }
00167
00168 char32_t UU(const std::string& s)
00169 {
00170   std::u32string s2 = Term::Private::utf8_to_utf32(s);
00171   if(s2.size() != 1) throw Term::Exception("U(s): s not a codepoint.");
00172   return s2[0];
00173 }
00174
00175 void Term::print_left_curly_bracket(Term::Window& scr, const std::size_t& x, const std::size_t& y1,
      const std::size_t& y2)
00176 {
00177   std::size_t h{y2 - y1 + 1};
00178   if(h == 1) { scr.set_char(x, y1, UU("]")); }
00179   else
```

```
00180   {
00181     scr.set_char(x, y1, UU("┐"));
00182     for(std::size_t j = y1 + 1; j <= y2 - 1; j++) { scr.set_char(x, j, UU("│")); }
00183     scr.set_char(x, y2, UU("┘"));
00184   }
00185 }
00186
00187 void Term::render(Term::Window& scr, const Model& m, const std::size_t& cols)
00188 {
00189   scr.clear();
00190   print_left_curly_bracket(scr, cols, 1, m.lines.size());
00191   scr.print_str(cols - 6, m.lines.size(), std::to_string(m.cursor_row) + "," +
      std::to_string(m.cursor_col));
00192   for(std::size_t j = 0; j < m.lines.size(); j++)
00193   {
00194     if(j == 0)
00195     {
00196       scr.fill_fg(1, j + 1, m.prompt_string.size(), m.lines.size(), Term::Color::Name::Green);
00197       scr.fill_style(1, j + 1, m.prompt_string.size(), m.lines.size(), Term::Style::Bold);
00198       scr.print_str(1, j + 1, m.prompt_string);
00199     }
00200     else
00201     {
00202       for(std::size_t i = 0; i < m.prompt_string.size() - 1; i++) { scr.set_char(i + 1, j + 1, '.'); }
00203     }
00204     scr.print_str(m.prompt_string.size() + 1, j + 1, m.lines[j]);
00205   }
00206   scr.set_cursor_pos(m.prompt_string.size() + m.cursor_col, m.cursor_row);
00207 }
00208
00209 std::string Term::prompt_multiline(const std::string& prompt_string, std::vector<std::string>&
      m_history, std::function<bool(std::string)>& iscomplete)
00210 {
00211   Term::Cursor cursor;
00212   Term::Screen screen(25, 80);
00213   bool         term_attached = Term::is_stdin_a_tty();
00214   if(is_stdin_a_tty())
00215   {
00216     cursor = cursor_position();
00217     screen = screen_size();
00218   }
00219
00220   Model m;
00221   m.prompt_string = prompt_string;
00222
00223   // Make a local copy of history that can be modified by the user. All
00224   // changes will be forgotten once a command is submitted.
00225   std::vector<std::string> history     = m_history;
00226   std::size_t              history_pos = history.size();
00227   history.push_back(concat(m.lines));  // Push back empty input
00228
00229   Term::Window scr(screen.columns(), 1);
00230   Term::Key    key;
00231   render(scr, m, screen.columns());
00232   std::cout « scr.render(1, cursor.row(), term_attached) « std::flush;
00233   bool not_complete = true;
00234   while(not_complete)
00235   {
00236     key = Term::read_event();
00237     if(key == Term::Key::NoKey) continue;
00238     if(key.isprint())
00239     {
00240       std::string before = m.lines[m.cursor_row - 1].substr(0, m.cursor_col - 1);
00241       std::string newchar;
00242       newchar.push_back(static_cast<char>(key));
00243       std::string after         = m.lines[m.cursor_row - 1].substr(m.cursor_col - 1);
00244       m.lines[m.cursor_row - 1] = before += newchar += after;
00245       m.cursor_col++;
00246     }
00247     else if(key == Key::Ctrl_D)
00248     {
00249       if(m.lines.size() == 1 && m.lines[m.cursor_row - 1].empty())
00250       {
00251         m.lines[m.cursor_row - 1].push_back(static_cast<char>(Key::Ctrl_D));
00252         std::cout « "\n" « std::flush;
00253         m_history.push_back(m.lines[0]);
00254         return m.lines[0];
00255       }
00256     }
00257     else
00258     {
00259       switch(key)
00260       {
00261         case Key::Enter:
00262           not_complete = !iscomplete(concat(m.lines));
00263           if(not_complete) key = Key(static_cast<Term::Key>(Term::MetaKey::Value::Alt +
      Term::Key::Enter));
```

```
00264           else
00265             break;
00266           CPP_TERMINAL_FALLTHROUGH;
00267         case Key::Backspace:
00268           if(m.cursor_col > 1)
00269           {
00270             std::string before        = m.lines[m.cursor_row - 1].substr(0, m.cursor_col - 2);
00271             std::string after         = m.lines[m.cursor_row - 1].substr(m.cursor_col - 1);
00272             m.lines[m.cursor_row - 1] = before + after;
00273             m.cursor_col--;
00274           }
00275           else if(m.cursor_col == 1 && m.cursor_row > 1)
00276           {
00277             m.cursor_col = m.lines[m.cursor_row - 2].size() + 1;
00278             m.lines[m.cursor_row - 2] += m.lines[m.cursor_row - 1];
00279             m.lines.erase(m.lines.begin() + static_cast<long>(m.cursor_row) - 1);
00280             m.cursor_row--;
00281           }
00282           break;
00283         case Key::Del:
00284           if(m.cursor_col <= m.lines[m.cursor_row - 1].size())
00285           {
00286             std::string before        = m.lines[m.cursor_row - 1].substr(0, m.cursor_col - 1);
00287             std::string after         = m.lines[m.cursor_row - 1].substr(m.cursor_col);
00288             m.lines[m.cursor_row - 1] = before + after;
00289           }
00290           break;
00291         case Key::ArrowLeft:
00292           if(m.cursor_col > 1) { m.cursor_col--; }
00293           break;
00294         case Key::ArrowRight:
00295           if(m.cursor_col <= m.lines[m.cursor_row - 1].size()) { m.cursor_col++; }
00296           break;
00297         case Key::Home: m.cursor_col = 1; break;
00298         case Key::End: m.cursor_col = m.lines[m.cursor_row - 1].size() + 1; break;
00299         case Key::ArrowUp:
00300           if(m.cursor_row == 1)
00301           {
00302             if(history_pos > 0)
00303             {
00304               history[history_pos] = concat(m.lines);
00305               history_pos--;
00306               m.lines      = split(history[history_pos]);
00307               m.cursor_row = m.lines.size();
00308               if(m.cursor_col > m.lines[m.cursor_row - 1].size() + 1) { m.cursor_col =
      m.lines[m.cursor_row - 1].size() + 1; }
00309               if(m.lines.size() > scr.get_h()) { scr.set_h(m.lines.size()); }
00310             }
00311           }
00312           else
00313           {
00314             m.cursor_row--;
00315             if(m.cursor_col > m.lines[m.cursor_row - 1].size() + 1) { m.cursor_col =
      m.lines[m.cursor_row - 1].size() + 1; }
00316           }
00317           break;
00318         case Key::ArrowDown:
00319           if(m.cursor_row == m.lines.size())
00320           {
00321             if(history_pos < history.size() - 1)
00322             {
00323               history[history_pos] = concat(m.lines);
00324               history_pos++;
00325               m.lines      = split(history[history_pos]);
00326               m.cursor_row = 1;
00327               if(m.cursor_col > m.lines[m.cursor_row - 1].size() + 1) { m.cursor_col =
      m.lines[m.cursor_row - 1].size() + 1; }
00328               if(m.lines.size() > scr.get_h()) { scr.set_h(m.lines.size()); }
00329             }
00330           }
00331           else
00332           {
00333             m.cursor_row++;
00334             if(m.cursor_col > m.lines[m.cursor_row - 1].size() + 1) { m.cursor_col =
      m.lines[m.cursor_row - 1].size() + 1; }
00335           }
00336           break;
00337         case Key::Ctrl_N:
00338         {
00339           std::string before        = m.lines[m.cursor_row - 1].substr(0, m.cursor_col - 1);
00340           std::string after         = m.lines[m.cursor_row - 1].substr(m.cursor_col - 1);
00341           m.lines[m.cursor_row - 1] = before;
00342           if(m.cursor_row < m.lines.size())
00343           {
00344             // Not at the bottom row, can't push back
00345             m.lines.insert(m.lines.begin() + static_cast<long>(m.cursor_row), after);
00346           }
```

```
00347           else { m.lines.push_back(after); }
00348           m.cursor_col = 1;
00349           m.cursor_row++;
00350           if(m.lines.size() > scr.get_h()) { scr.set_h(m.lines.size()); }
00351           break;
00352         }
00353         default: break;
00354       }
00355     }
00356     render(scr, m, screen.columns());
00357     std::cout « scr.render(1, cursor.row(), term_attached) « std::flush;
00358     if(cursor.row() + (int)scr.get_h() - 1 > screen.rows())
00359     {
00360       cursor.setRow(static_cast<std::uint16_t>(static_cast<long>(screen.rows()) -
      (static_cast<long>(scr.get_h()) - 1)));
00361       std::cout « scr.render(1, cursor.row(), term_attached) « std::flush;
00362     }
00363   }
00364   std::string line_skips;
00365   for(std::size_t i = 0; i <= m.lines.size() - m.cursor_row; i++) { line_skips += "\n"; }
00366   std::cout « line_skips « std::flush;
00367   m_history.push_back(concat(m.lines));
00368   return concat(m.lines);
00369 }
```

9.111 cpp-terminal/prompt.hpp File Reference

```
#include "cpp-terminal/terminal.hpp"
#include "cpp-terminal/window.hpp"
#include <functional>
```

Classes

- class Term::Model

Namespaces

- namespace Term

Enumerations

- enum class Term::Result {
  Term::Yes , Term::No , Term::Error , Term::None ,
  Term::Abort , Term::Invalid }
- enum class Term::Result_simple { Term::Yes , Term::No , Term::Abort }

Functions

- Result Term::prompt (const std::string &message, const std::string &first_option, const std::string &second↩
  _option, const std::string &prompt_indicator, bool)

  A simple yes/no prompt, requires the user to press the ENTER key to continue.
- Result_simple Term::prompt_simple (const std::string &message)

  The most simple prompt possible, requires the user to press enter to continue.
- std::string Term::concat (const std::vector< std::string > &)
- std::vector< std::string > Term::split (const std::string &)
- void Term::print_left_curly_bracket (Term::Window &, const std::size_t &, const std::size_t &, const std::size↩
  _t &)
- void Term::render (Term::Window &, const Model &, const std::size_t &)
- std::string Term::prompt_multiline (const std::string &, std::vector< std::string > &, std::function< bool(std↩
  ::string)> &)

9.112 prompt.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/terminal.hpp"
00013 #include "cpp-terminal/window.hpp"
00014
00015 #include <functional>
00016
00017 namespace Term
00018 {
00019
00020 // indicates the results of prompt_blocking() and prompt_non_blocking
00021 enum class Result
00022 {
00023   Yes,
00024   No,
00025   Error,
00026   None,
00027   Abort,
00028   Invalid
00029 };
00030
00041 Result prompt(const std::string& message, const std::string& first_option, const std::string&
      second_option, const std::string& prompt_indicator, bool);
00042
00043 // indicates the results of prompt_simple()
00044 enum class Result_simple
00045 {
00046
00047   Yes,
00048   No,
00049   Abort
00050 };
00051
00059 Result_simple prompt_simple(const std::string& message);
00060
00061 /* Multiline prompt */
00062
00063 // This model contains all the information about the state of the prompt in an
00064 // abstract way, irrespective of where or how it is rendered.
00065 class Model
00066 {
00067 public:
00068   std::string              prompt_string;  // The string to show as the prompt
00069   std::vector<std::string> lines{""};      // The current input string in the prompt as a vector of
      lines, without '\n' at the end.
00070   // The current cursor position in the "input" string, starting from (1,1)
00071   std::size_t              cursor_col{1};
00072   std::size_t              cursor_row{1};
00073 };
00074
00075 std::string concat(const std::vector<std::string>&);
00076
00077 std::vector<std::string> split(const std::string&);
00078
00079 void print_left_curly_bracket(Term::Window&, const std::size_t&, const std::size_t&, const
      std::size_t&);
00080
00081 void render(Term::Window&, const Model&, const std::size_t&);
00082
00083 std::string prompt_multiline(const std::string&, std::vector<std::string>&,
      std::function<bool(std::string)>&);
00084 }  // namespace Term
```

9.113 cpp-terminal/screen.hpp File Reference

```
#include <cstdint>
#include <string>
#include <utility>
```

Classes

- class Term::Screen

Namespaces

- namespace Term

Functions

- std::string Term::clear_screen ()
- std::string Term::screen_save ()
- std::string Term::screen_load ()
- Screen Term::screen_size ()

9.114 screen.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstdint>
00013 #include <string>
00014 #include <utility>
00015
00016 namespace Term
00017 {
00018
00019 class Screen
00020 {
00021 public:
00022   Screen() = default;
00023   Screen(const std::size_t& rows, const std::size_t& columns);
00024   std::size_t rows() const;
00025   std::size_t columns() const;
00026   bool        empty() const;
00027   bool        operator==(const Term::Screen& screen) const;
00028   bool        operator!=(const Term::Screen& screen) const;
00029
00030 private:
00031   std::pair<std::size_t, std::size_t> m_size;
00032 };
00033
00034 // clear the screen
00035 std::string clear_screen();
00036 // save the current terminal state
00037 std::string screen_save();
00038 // load a previously saved terminal state
00039 std::string screen_load();
00040 // get the terminal size (row, column) / (Y, X)
00041 Screen      screen_size();
00042
00043 }  // namespace Term
```

9.115 cpp-terminal/stream.cpp File Reference

```
#include "cpp-terminal/stream.hpp"
```

9.116 stream.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/stream.hpp"
00011
00012 Term::TIstream::TIstream(const Term::Buffer::Type& type, const std::streamsize& size) : m_buffer(type,
      size), m_stream(&m_buffer) {}
00013
00014 Term::TIstream::~TIstream() { m_stream.clear(); }
00015
00016 std::streambuf* Term::TIstream::rdbuf() const { return const_cast<Term::Buffer*>(&m_buffer); }
00017
00018 Term::TOstream::TOstream(const Term::Buffer::Type& type, const std::streamsize& size) : m_buffer(type,
      size), m_stream(&m_buffer) {}
00019
00020 Term::TOstream::~TOstream() { m_stream.flush(); }
```

9.117 cpp-terminal/stream.hpp File Reference

```
#include "cpp-terminal/buffer.hpp"
#include <istream>
#include <ostream>
```

Classes

- class Term::TIstream
- class Term::TOstream

Namespaces

- namespace Term

9.118 stream.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/buffer.hpp"
00013
00014 #include <istream>
00015 #include <ostream>
00016
00017 namespace Term
00018 {
00019
00020 class TIstream
00021 {
```

```
00022 public:
00023   explicit TIstream(const Term::Buffer::Type& type = Term::Buffer::Type::LineBuffered, const
      std::streamsize& size = BUFSIZ);
00024   TIstream(const TIstream&)            = delete;
00025   TIstream(TIstream&& other)           = delete;
00026   TIstream& operator=(TIstream&&)      = delete;
00027   TIstream& operator=(const TIstream&) = delete;
00028   ~TIstream();
00029   std::streambuf*                rdbuf() const;
00030   template<typename T> TIstream& operator»(T& t)
00031   {
00032     m_stream » t;
00033     return *this;
00034   }
00035
00036 private:
00037   Term::Buffer m_buffer;
00038   std::istream m_stream;
00039 };
00040
00041 class TOstream
00042 {
00043 public:
00044   explicit TOstream(const Term::Buffer::Type& type = Term::Buffer::Type::LineBuffered, const
      std::streamsize& size = BUFSIZ);
00045   ~TOstream();
00046   TOstream(const TOstream&)                                  = delete;
00047   TOstream(TOstream&&)                                       = delete;
00048   TOstream&                      operator=(TOstream&&)      = delete;
00049   TOstream&                      operator=(const TOstream&) = delete;
00050   template<typename T> TOstream& operator«(const T& t)
00051   {
00052     m_stream « t;
00053     return *this;
00054   }
00055   TOstream& operator«(std::ostream& (*t)(std::ostream&))
00056   {
00057     m_stream « t;
00058     return *this;
00059   }
00060
00061 private:
00062   Term::Buffer m_buffer;
00063   std::ostream m_stream;
00064 };
00065
00066 }  // namespace Term
```

9.119 cpp-terminal/style.cpp File Reference

```
#include "cpp-terminal/style.hpp"
```

9.120 style.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/style.hpp"
00011
00012 std::string Term::style(const Term::Style& style)
00013 {
00014
      //https://unix.stackexchange.com/questions/212933/background-color-whitespace-when-end-of-the-terminal-reached
00015   std::string ret{"\u001b[" + std::to_string(static_cast<std::uint8_t>(style)) + 'm'};
00016   if(style == Term::Style::DefaultBackgroundColor) { ret += "\u001b[K"; }
00017   return ret;
00018 }
```

9.121 cpp-terminal/style.hpp File Reference

```
#include "cpp-terminal/iostream.hpp"
#include <cstdint>
#include <string>
```

Namespaces

- namespace Term

Enumerations

- enum class Term::Style : std::uint8_t {
  Term::Reset = 0 , Term::Bold = 1 , Term::Dim = 2 , Term::Italic = 3 ,
  Term::Underline = 4 , Term::Blink = 5 , Term::BlinkRapid = 6 , Term::Reversed = 7 ,
  Term::Conceal = 8 , Term::Crossed = 9 , Term::Font0 = 10 , Term::ResetFont = 10 ,
  Term::Font1 = 11 , Term::Font2 = 12 , Term::Font3 = 13 , Term::Font4 = 14 ,
  Term::Font5 = 15 , Term::Font6 = 16 , Term::Font7 = 17 , Term::Font8 = 18 ,
  Term::Font9 = 19 , Term::Font10 = 20 , Term::DoublyUnderlinedOrNotBold = 21 , Term::ResetBold = 22 ,
  Term::ResetDim = 22 , Term::ResetItalic = 23 , Term::ResetUnderline = 24 , Term::ResetBlink = 25 ,
  Term::ResetBlinkRapid = 25 , Term::ResetReversed = 27 , Term::ResetConceal = 28 , Term::ResetCrossed
  = 29 ,
  Term::DefaultForegroundColor = 39 , Term::DefaultBackgroundColor = 49 , Term::Frame = 51 , Term::Encircle
  = 52 ,
  Term::Overline = 53 , Term::ResetFrame = 54 , Term::ResetEncircle = 54 , Term::ResetOverline = 55 ,
  Term::DefaultUnderlineColor = 59 , Term::BarRight = 60 , Term::DoubleBarRight = 61 , Term::BarLeft = 62 ,
  Term::DoubleBarLeft = 63 , Term::StressMarking = 64 , Term::ResetBar = 65 , Term::Superscript = 73 ,
  Term::Subscript = 74 , Term::ResetSuperscript = 75 , Term::ResetSubscript = 75 }

Functions

- std::string Term::style (const Term::Style &style)
- template<class Stream >
  Stream & Term::operator<< (Stream &stream, const Term::Style &style_type)
- Term::TOstream & Term::operator<< (Term::TOstream &term, const Term::Style &style_type)

9.122 style.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/iostream.hpp"
00013
00014 #include <cstdint>
00015 #include <string>
00016
00017 namespace Term
00018 {
```

```
00019
00020 /*
00021  * Styles for text in the terminal
00022  */
00023 enum class Style : std::uint8_t
00024 {
00025
00026   Reset      = 0,
00027   Bold       = 1,
00028   Dim        = 2,
00029   Italic     = 3,
00030   Underline  = 4,
00031   Blink      = 5,
00032   BlinkRapid = 6,
00033   Reversed   = 7,
00034   Conceal    = 8,
00035   Crossed    = 9,
00036
00037   // different fonts
00038   Font0     = 10,
00039   ResetFont = 10,
00040   Font1     = 11,
00041   Font2     = 12,
00042   Font3     = 13,
00043   Font4     = 14,
00044   Font5     = 15,
00045   Font6     = 16,
00046   Font7     = 17,
00047   Font8     = 18,
00048   Font9     = 19,
00049   Font10    = 20,
00050
00051   // Double-underline per ECMA-48,[5] 8.3.117 but instead disables bold intensity on several
      terminals,
00052   // including in the Linux kernel's console before version 4.17
00053   DoublyUnderlinedOrNotBold = 21,
00054
00055   // resets corresponding styles
00056   ResetBold       = 22,
00057   ResetDim        = 22,
00058   ResetItalic     = 23,
00059   ResetUnderline  = 24,
00060   ResetBlink      = 25,
00061   ResetBlinkRapid = 25,
00062   ResetReversed   = 27,
00063   ResetConceal    = 28,
00064   ResetCrossed    = 29,
00065
00066   // sets the foreground and background color to the implementation defined colors
00067   DefaultForegroundColor = 39,
00068   DefaultBackgroundColor = 49,
00069
00070   Frame         = 51,
00071   Encircle      = 52,
00072   Overline      = 53,  // draw a line over the text, barely supported
00073   ResetFrame    = 54,
00074   ResetEncircle = 54,
00075   ResetOverline = 55,
00076
00077   // sets the underline color to the implementation defined colors
00078   DefaultUnderlineColor = 59,
00079
00080   BarRight       = 60,
00081   DoubleBarRight = 61,
00082   BarLeft        = 62,
00083   DoubleBarLeft  = 63,
00084   StressMarking  = 64,
00085
00086   ResetBar = 65,  // resets 60 - 64 inclusive
00087
00088   Superscript      = 73,  // only implemented in mintty
00089   Subscript        = 74,  // only implemented in mintty
00090   ResetSuperscript = 75,  // only implemented in mintty
00091   ResetSubscript   = 75
00092 };
00093
00094 std::string style(const Term::Style& style);
00095
00096 template<class Stream> Stream& operator«(Stream& stream, const Term::Style& style_type) { return
      stream « style(style_type); }
00097 // unabigify operator overload
00098 inline Term::TOstream&         operator«(Term::TOstream& term, const Term::Style& style_type) { return
      term « style(style_type); }
00099
00100 }  // namespace Term
```

9.123 cpp-terminal/terminal.cpp File Reference

```
#include "cpp-terminal/terminal.hpp"
#include <array>
```

9.124 terminal.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/terminal.hpp"
00011
00012 #include <array>
00013
00014 namespace
00015 {
00016 std::array<char, sizeof(Term::Terminal)> terminal_buffer;
      //NOLINT(fuchsia-statically-constructed-objects)
00017 }  // namespace
00018
00019 Term::Terminal& Term::terminal = reinterpret_cast<Term::Terminal&>(::terminal_buffer);
      //NOLINT(cppcoreguidelines-pro-type-reinterpret-cast)
00020
00021 std::string Term::terminal_title(const std::string& title) { return "\u001b]0;" + title + '\a'; }
00022
00023 std::string Term::clear_buffer() { return "\u001b[3J"; }
```

9.125 cpp-terminal/terminal.hpp File Reference

```
#include "cpp-terminal/terminal_impl.hpp"
#include "cpp-terminal/terminal_initializer.hpp"
#include <string>
```

Namespaces

- namespace Term

Functions

- std::string Term::terminal_title (const std::string &title)
- std::string Term::clear_buffer ()

Variables

- Term::Terminal & Term::terminal = reinterpret_cast<Term::Terminal&>(::terminal_buffer)

9.126 terminal.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/terminal_impl.hpp"
00013 #include "cpp-terminal/terminal_initializer.hpp"
00014
00015 #include <string>
00016
00017 namespace Term
00018 {
00019
00020 static const TerminalInitializer terminal_initializer;
      //NOLINT(cert-err58-cpp,fuchsia-statically-constructed-objects)
00021 extern Term::Terminal&           terminal;
00022
00023 // change the title of the terminal, only supported by a few terminals
00024 std::string terminal_title(const std::string& title);
00025 // clear the screen and the scroll-back buffer
00026 std::string clear_buffer();
00027
00028 }  // namespace Term
```

9.127 cpp-terminal/terminal_impl.hpp File Reference

```
#include "cpp-terminal/options.hpp"
#include "cpp-terminal/terminal_initializer.hpp"
#include <cstddef>
```

Classes

- class Term::Terminal

Namespaces

- namespace Term

9.128 terminal_impl.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/options.hpp"
00013 #include "cpp-terminal/terminal_initializer.hpp"
00014
00015 #include <cstddef>
00016
```

```
00017 namespace Term
00018 {
00019
00020 class Terminal
00021 {
00022 public:
00023   ~Terminal() noexcept;
00024   Terminal() noexcept;
00025   Terminal(const Terminal&)                                    = delete;
00026   Terminal(Terminal&&)                                         = delete;
00027   Terminal&                      operator=(Terminal&&)       = delete;
00028   Terminal&                      operator=(const Terminal&) = delete;
00029   template<typename... Args> void setOptions(const Args&&... args)
00030   {
00031     m_options = {args...};
00032     applyOptions();
00033   }
00034   Term::Options getOptions() const noexcept;
00035
00036 private:
00040   static void store_and_restore() noexcept;
00041
00046   void setMode() const;
00047
00048   void setOptions();
00049   void applyOptions() const;
00050
00051   static std::size_t setMouseEvents();
00052   static std::size_t unsetMouseEvents();
00053   static std::size_t setFocusEvents();
00054   static std::size_t unsetFocusEvents();
00055
00056   static void   set_unset_utf8();
00057   Term::Options m_options;
00058 };
00059
00060 }  // namespace Term
```

9.129 cpp-terminal/terminal_initializer.cpp File Reference

```
#include "cpp-terminal/terminal_initializer.hpp"
#include "cpp-terminal/private/exception.hpp"
#include "cpp-terminal/private/file_initializer.hpp"
#include "cpp-terminal/terminal.hpp"
#include <new>
```

9.130 terminal_initializer.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/terminal_initializer.hpp"
00011
00012 #include "cpp-terminal/private/exception.hpp"
00013 #include "cpp-terminal/private/file_initializer.hpp"
00014 #include "cpp-terminal/terminal.hpp"
00015
00016 #include <new>
00017
00018 std::size_t Term::TerminalInitializer::m_counter{0};
00019
00020 Term::TerminalInitializer::TerminalInitializer() noexcept
00021 try
00022 {
00023   if(0 == m_counter)
00024   {
00025     static const Private::FileInitializer files_init;
```

```
00026     new(&Term::terminal) Terminal();
00027   }
00028   ++m_counter;
00029 }
00030 catch(...)
00031 {
00032   ExceptionHandler(Private::ExceptionDestination::StdErr);
00033 }
00034
00035 Term::TerminalInitializer::~TerminalInitializer() noexcept
00036 try
00037 {
00038   --m_counter;
00039   if(0 == m_counter) { (&Term::terminal)->~Terminal(); }
00040 }
00041 catch(...)
00042 {
00043   ExceptionHandler(Private::ExceptionDestination::StdErr);
00044 }
```

9.131 cpp-terminal/terminal_initializer.hpp File Reference

```
#include <cstddef>
```

Classes

- class Term::TerminalInitializer

Namespaces

- namespace Term

9.132 terminal_initializer.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstddef>
00013
00014 namespace Term
00015 {
00016
00017 class TerminalInitializer
00018 {
00019 public:
00020   ~TerminalInitializer() noexcept;
00021   TerminalInitializer() noexcept;
00022   TerminalInitializer(const TerminalInitializer&)            = delete;
00023   TerminalInitializer(TerminalInitializer&&)                 = delete;
00024   TerminalInitializer& operator=(TerminalInitializer&&)      = delete;
00025   TerminalInitializer& operator=(const TerminalInitializer&) = delete;
00026
00027 private:
00028   static std::size_t m_counter;
00029 };
00030
00031 }  // namespace Term
```

9.133 cpp-terminal/terminfo.hpp File Reference

```
#include <array>
#include <cstdint>
#include <string>
```

Classes

- class Term::Terminfo

Namespaces

- namespace Term

9.134 terminfo.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <array>
00013 #include <cstdint>
00014 #include <string>
00015
00016 namespace Term
00017 {
00018
00019 class Terminfo
00020 {
00021 public:
00022   // indicates the color mode (basically the original color resolution)
00023   // also used to manually override the original color resolution
00024   enum class ColorMode : std::uint8_t
00025   {
00026     Unset,
00027     // no color was used
00028     NoColor,
00029     // a 3bit color was used
00030     Bit3,
00031     // a 4bit color was used
00032     Bit4,
00033     // a 8bit color was used
00034     Bit8,
00035     // a 24bit (RGB) color was used
00036     Bit24
00037   };
00038   enum class Bool : std::uint8_t
00039   {
00040     UTF8 = 0,
00041     Legacy,
00042     ControlSequences,
00043   };
00044   enum class String : std::uint8_t
00045   {
00046     TermEnv,
00047     TermName,
00048     TermVersion,
00049   };
00050   enum class Integer : std::uint8_t
00051   {
00052
00053   };
```

```
00054
00055   static bool          get(const Term::Terminfo::Bool& key);
00056   static std::uint32_t get(const Term::Terminfo::Integer& key);
00057   static std::string   get(const Term::Terminfo::String& key);
00058
00059 private:
00060   static const constexpr std::size_t BoolNumber{3};
00061   static const constexpr std::size_t StringNumber{3};
00062   static const constexpr std::size_t IntegerNumber{0};
00063
00064 public:
00065   using Booleans = std::array<bool, BoolNumber>;
00066   using Strings  = std::array<std::string, StringNumber>;
00067   using Integers = std::array<std::uint32_t, IntegerNumber>;
00068
00069   Terminfo();
00070
00071   static ColorMode getColorMode();
00072
00073 private:
00074   static void check();
00075   static void checkTermEnv();
00076   static void checkTerminalName();
00077   static void checkTerminalVersion();
00078   static void checkColorMode();
00079   static void checkUTF8();
00080   static void checkLegacy();
00081   static void checkControlSequences();
00082
00083   static void set(const Term::Terminfo::Bool& key, const bool& value);
00084   static void set(const Term::Terminfo::Integer& key, const std::uint32_t& value);
00085   static void set(const Term::Terminfo::String& key, const std::string& value);
00086
00087   static ColorMode m_colorMode;
00088   static Booleans  m_booleans;
00089   static Integers  m_integers;
00090   static Strings   m_strings;
00091 };
00092
00093 }  // namespace Term
```

9.135 cpp-terminal/tty.hpp File Reference

Namespaces

- namespace Term

Functions

- bool Term::is_stdin_a_tty ()

  *Check if **stdin** is a **tty**.*
- bool Term::is_stdout_a_tty ()

  *Check if **stdout** is a **tty**.*
- bool Term::is_stderr_a_tty ()

  *Check if **stderr** is a **tty**.*

9.136 tty.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
```

```
00011
00012 namespace Term
00013 {
00014
00021 bool is_stdin_a_tty();
00022
00029 bool is_stdout_a_tty();
00030
00037 bool is_stderr_a_tty();
00038
00039 }  // namespace Term
```

9.137 cpp-terminal/version.hpp File Reference

```
#include <cstdint>
#include <string>
```

Namespaces

- namespace Term
- namespace Term::Version

Functions

- std::uint16_t Term::Version::major () noexcept

  Major version of cpp-terminal.
- std::uint16_t Term::Version::minor () noexcept

  Minor version of cpp-terminal.
- std::uint16_t Term::Version::patch () noexcept

  Patch version of cpp-terminal.
- std::string Term::Version::string () noexcept

  String version of cpp-terminal.
- std::string Term::homepage () noexcept

  Homepage of cpp-terminal.

9.138 version.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include <cstdint>
00013 #include <string>
00014
00015 namespace Term
00016 {
00017 namespace Version
00018 {
00019
00025 std::uint16_t major() noexcept;
00026
00032 std::uint16_t minor() noexcept;
00033
```

```
00039 std::uint16_t patch() noexcept;
00040
00046 std::string string() noexcept;
00047
00048 }  // namespace Version
00049
00055 std::string homepage() noexcept;
00056
00057 }  // namespace Term
```

9.139 cpp-terminal/window.cpp File Reference

```
#include "cpp-terminal/window.hpp"
#include "cpp-terminal/color.hpp"
#include "cpp-terminal/cursor.hpp"
#include "cpp-terminal/exception.hpp"
#include "cpp-terminal/private/conversion.hpp"
#include "cpp-terminal/private/unicode.hpp"
#include "cpp-terminal/prompt.hpp"
#include "cpp-terminal/terminal.hpp"
#include "cpp-terminal/terminfo.hpp"
#include <cstddef>
```

Namespaces

- namespace Term

9.140 window.cpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #include "cpp-terminal/window.hpp"
00011
00012 #include "cpp-terminal/color.hpp"
00013 #include "cpp-terminal/cursor.hpp"
00014 #include "cpp-terminal/exception.hpp"
00015 #include "cpp-terminal/private/conversion.hpp"
00016 #include "cpp-terminal/private/unicode.hpp"
00017 #include "cpp-terminal/prompt.hpp"
00018 #include "cpp-terminal/terminal.hpp"
00019 #include "cpp-terminal/terminfo.hpp"
00020
00021 #include <cstddef>
00022
00023 namespace Term
00024 {
00025
00026 Term::Window::Window(const std::size_t& columns, const std::size_t& rows) : m_window({rows, columns})
      { clear(); }
00027
00028 char32_t Term::Window::get_char(const std::size_t& column, const std::size_t& row) { return
      m_chars[index(column, row)]; }
00029
00030 bool Term::Window::get_fg_reset(const std::size_t& column, const std::size_t& row) { return
      m_fg_reset[index(column, row)]; }
00031
00032 bool Term::Window::get_bg_reset(const std::size_t& column, const std::size_t& row) { return
      m_bg_reset[index(column, row)]; }
00033
```

```
00034 Term::Color Term::Window::get_fg(const std::size_t& column, const std::size_t& row) { return
      m_fg[index(column, row)]; }
00035
00036 Term::Color Term::Window::get_bg(const std::size_t& column, const std::size_t& row) { return
      m_bg[index(column, row)]; }
00037
00038 Term::Style Term::Window::get_style(const std::size_t& column, const std::size_t& row) { return
      m_style[index(column, row)]; }
00039
00040 std::size_t Term::Window::get_w() const { return m_window.columns(); }
00041
00042 std::size_t Term::Window::get_h() const { return m_window.rows(); }
00043
00044 void Term::Window::set_char(const std::size_t& column, const std::size_t& row, const char32_t&
      character)
00045 {
00046   if(insideWindow(column, row)) { m_chars[index(column, row)] = character; }
00047   else { throw Term::Exception("set_char(): (x,y) out of bounds"); }
00048 }
00049
00050 void Term::Window::set_fg_reset(const std::size_t& column, const std::size_t& row)
00051 {
00052   m_fg_reset[index(column, row)] = true;
00053   m_fg[index(column, row)]       = Term::Color::Name::Default;
00054 }
00055
00056 void Term::Window::set_bg_reset(const std::size_t& column, const std::size_t& row)
00057 {
00058   m_bg_reset[index(column, row)] = true;
00059   m_bg[index(column, row)]       = Term::Color::Name::Default;
00060 }
00061
00062 void Term::Window::set_fg(const std::size_t& column, const std::size_t& row, const Color& color)
00063 {
00064   m_fg_reset[index(column, row)] = false;
00065   m_fg[index(column, row)]       = color;
00066 }
00067
00068 void Term::Window::set_bg(const std::size_t& column, const std::size_t& row, const Color& color)
00069 {
00070   m_bg_reset[index(column, row)] = false;
00071   m_bg[index(column, row)]       = color;
00072 }
00073
00074 void Term::Window::set_style(const std::size_t& column, const std::size_t& row, const Style& style) {
      m_style[index(column, row)] = style; }
00075
00076 void Term::Window::set_cursor_pos(const std::size_t& column, const std::size_t& row) { m_cursor =
      {row, column}; }
00077
00078 void Term::Window::set_h(const std::size_t& new_h)
00079 {
00080   if(new_h == m_window.rows()) { return; }
00081   if(new_h > m_window.rows())
00082   {
00083     const std::size_t dc = (new_h - m_window.rows()) * m_window.columns();
00084     m_chars.insert(m_chars.end(), dc, ' ');
00085     m_fg_reset.insert(m_fg_reset.end(), dc, true);
00086     m_bg_reset.insert(m_bg_reset.end(), dc, true);
00087     m_fg.insert(m_fg.end(), dc, {0, 0, 0});
00088     m_bg.insert(m_bg.end(), dc, {0, 0, 0});
00089     m_style.insert(m_style.end(), dc, Style::Reset);
00090     m_window = {m_window.columns(), new_h};
00091   }
00092   else { throw Term::Exception("Shrinking height not supported."); }
00093 }
00094
00095 void Term::Window::print_str(const std::size_t& x, const std::size_t& y, const std::string& s, const
      std::size_t& indent, bool move_cursor)
00096 {
00097   std::u32string s2   = Private::utf8_to_utf32(s);
00098   std::size_t    xpos = x;
00099   std::size_t    ypos = y;
00100   for(char32_t i: s2)
00101   {
00102     if(i == U'\n')
00103     {
00104       xpos = x + indent;
00105       ypos++;
00106       if(insideWindow(xpos, ypos))
00107       {
00108         for(std::size_t j = 0; j < indent; ++j) { set_char(x + j, ypos, '.'); }
00109       }
00110       else { return; }
00111     }
00112     else
00113     {
```

```
00114       if(insideWindow(xpos, ypos)) { set_char(xpos, y, i); }
00115       else { return; }
00116       ++xpos;
00117     }
00118   }
00119   if(move_cursor) { m_cursor = {ypos, xpos}; }
00120 }
00121
00122 void Term::Window::fill_fg(const std::size_t& x1, const std::size_t& y1, const std::size_t& x2, const
      std::size_t& y2, const Color& rgb)
00123 {
00124   for(std::size_t j = y1; j <= y2; ++j)
00125   {
00126     for(std::size_t i = x1; i <= x2; ++i) { set_fg(i, j, rgb); }
00127   }
00128 }
00129
00130 void Term::Window::fill_bg(const std::size_t& x1, const std::size_t& y1, const std::size_t& x2, const
      std::size_t& y2, const Color& rgb)
00131 {
00132   for(std::size_t j = y1; j <= y2; ++j)
00133   {
00134     for(std::size_t i = x1; i <= x2; ++i) { set_bg(i, j, rgb); }
00135   }
00136 }
00137
00138 void Term::Window::fill_style(const std::size_t& x1, const std::size_t& y1, const std::size_t& x2,
      const std::size_t& y2, const Style& color)
00139 {
00140   for(std::size_t j = y1; j <= y2; ++j)
00141   {
00142     for(std::size_t i = x1; i <= x2; ++i) { set_style(i, j, color); }
00143   }
00144 }
00145
00146 void Term::Window::print_border() { print_rect(1, 1, m_window.columns(), m_window.rows()); }
00147
00148 void Term::Window::print_rect(const std::size_t& x1, const std::size_t& y1, const std::size_t& x2,
      const std::size_t& y2)
00149 {
00150   std::u32string border = Private::utf8_to_utf32("│─┌┐└┘");
00151   if(Term::Terminfo::get(Term::Terminfo::Bool::UTF8))
00152   {
00153     for(std::size_t j = y1 + 1; j <= (y2 - 1); ++j)
00154     {
00155       set_char(x1, j, border[0]);
00156       set_char(x2, j, border[0]);
00157     }
00158     for(std::size_t i = x1 + 1; i <= (x2 - 1); ++i)
00159     {
00160       set_char(i, y1, border[1]);
00161       set_char(i, y2, border[1]);
00162     }
00163     set_char(x1, y1, border[2]);
00164     set_char(x2, y1, border[3]);
00165     set_char(x1, y2, border[4]);
00166     set_char(x2, y2, border[5]);
00167   }
00168   else
00169   {
00170     for(std::size_t j = y1 + 1; j <= (y2 - 1); ++j)
00171     {
00172       set_char(x1, j, '|');
00173       set_char(x2, j, '|');
00174     }
00175     for(std::size_t i = x1 + 1; i <= (x2 - 1); ++i)
00176     {
00177       set_char(i, y1, '-');
00178       set_char(i, y2, '-');
00179     }
00180     set_char(x1, y1, '+');
00181     set_char(x2, y1, '+');
00182     set_char(x1, y2, '+');
00183     set_char(x2, y2, '+');
00184   }
00185 }
00186
00187 void Term::Window::clear()
00188 {
00189   const std::size_t area{m_window.rows() * m_window.columns()};
00190   m_style.assign(area, Style::Reset);
00191   m_bg_reset.assign(area, true);
00192   m_bg.assign(area, Term::Color::Name::Default);
00193   m_fg_reset.assign(area, true);
00194   m_fg.assign(area, Term::Color::Name::Default);
00195   m_chars.assign(area, ' ');
00196 }
```

```
00197
00198 std::string Term::Window::render(const std::size_t& x0, const std::size_t& y0, bool term)
00199 {
00200   std::string out;
00201   if(term) { out.append(cursor_off()); }
00202   Color current_fg       = Term::Color::Name::Default;
00203   Color current_bg       = Term::Color::Name::Default;
00204   bool  current_fg_reset = true;
00205   bool  current_bg_reset = true;
00206   Style current_style    = Style::Reset;
00207   for(std::size_t j = 1; j <= m_window.rows(); ++j)
00208   {
00209     if(term) { out.append(cursor_move(y0 + j - 1, x0)); }
00210     for(std::size_t i = 1; i <= m_window.columns(); ++i)
00211     {
00212       bool update_fg       = false;
00213       bool update_bg       = false;
00214       bool update_fg_reset = false;
00215       bool update_bg_reset = false;
00216       bool update_style    = false;
00217       if(current_fg_reset != get_fg_reset(i, j))
00218       {
00219         current_fg_reset = get_fg_reset(i, j);
00220         if(current_fg_reset)
00221         {
00222           update_fg_reset = true;
00223           current_fg      = {255, 255, 255};
00224         }
00225       }
00226
00227       if(current_bg_reset != get_bg_reset(i, j))
00228       {
00229         current_bg_reset = get_bg_reset(i, j);
00230         if(current_bg_reset)
00231         {
00232           update_bg_reset = true;
00233           current_bg      = {255, 255, 255};
00234         }
00235       }
00236
00237       if(!current_fg_reset)
00238       {
00239         if(!(current_fg == get_fg(i, j)))
00240         {
00241           current_fg = get_fg(i, j);
00242           update_fg  = true;
00243         }
00244       }
00245
00246       if(!current_fg_reset)
00247       {
00248         if(!(current_bg == get_bg(i, j)))
00249         {
00250           current_bg = get_bg(i, j);
00251           update_bg  = true;
00252         }
00253       }
00254       if(current_style != get_style(i, j))
00255       {
00256         current_style = get_style(i, j);
00257         update_style  = true;
00258         if(current_style == Style::Reset)
00259         {
00260           // style::reset: reset fg and bg colors too, we have to
00261           // set them again if they are non-default, but if fg or
00262           // bg colors are reset, we do not update them, as
00263           // style::reset already did that.
00264           update_fg = !current_fg_reset;
00265           update_bg = !current_bg_reset;
00266         }
00267       }
00268       // Set style first, as style::reset will reset colors too
00269       if(update_style) { out.append(style(get_style(i, j))); }
00270       if(update_fg_reset) { out.append(color_fg(Term::Color::Name::Default)); }
00271       else if(update_fg)
00272       {
00273         const Term::Color color_tmp = get_fg(i, j);
00274         out.append(color_fg(color_tmp));
00275       }
00276
00277       if(update_bg_reset) { out.append(color_bg(Term::Color::Name::Default)); }
00278       else if(update_bg)
00279       {
00280         const Term::Color color_tmp = get_bg(i, j);
00281         out.append(color_bg(color_tmp));
00282       }
00283       out.append(Private::utf32_to_utf8(get_char(i, j)));
```

```
00284     }
00285     if(j < m_window.rows()) { out.append("\n"); }
00286   }
00287   if(!current_fg_reset) { out.append(color_fg(Term::Color::Name::Default)); }
00288   if(!current_bg_reset) { out.append(color_bg(Term::Color::Name::Default)); }
00289   if(current_style != Style::Reset) { out.append(style(Style::Reset)); }
00290   if(term)
00291   {
00292     out.append(cursor_move(y0 + (m_cursor.row() - 1), x0 + (m_cursor.column() - 1)));
00293     out.append(cursor_on());
00294   }
00295   return out;
00296 }
00297
00298 std::size_t Term::Window::index(const std::size_t& column, const std::size_t& row) const
00299 {
00300   if(!insideWindow(column, row)) { throw Term::Exception("Cursor out of range"); }
00301   return ((row - 1) * m_window.columns()) + (column - 1);
00302 }
00303
00304 bool Term::Window::insideWindow(const std::size_t& column, const std::size_t& row) const { return
      (column >= 1) && (row >= 1) && (column <= m_window.columns()) && (row <= m_window.rows()); }
00305 }  // namespace Term
```

9.141 cpp-terminal/window.hpp File Reference

```
#include "cpp-terminal/color.hpp"
#include "cpp-terminal/cursor.hpp"
#include "cpp-terminal/screen.hpp"
#include "cpp-terminal/style.hpp"
#include <cstddef>
#include <vector>
```

Classes

- class Term::Window

  Represents a rectangular window, as a 2D array of characters and their attributes.

Namespaces

- namespace Term

9.142 window.hpp

Go to the documentation of this file.

```
00001 /*
00002 * cpp-terminal
00003 * C++ library for writing multi-platform terminal applications.
00004 *
00005 * SPDX-FileCopyrightText: 2019-2024 cpp-terminal
00006 *
00007 * SPDX-License-Identifier: MIT
00008 */
00009
00010 #pragma once
00011
00012 #include "cpp-terminal/color.hpp"
00013 #include "cpp-terminal/cursor.hpp"
00014 #include "cpp-terminal/screen.hpp"
00015 #include "cpp-terminal/style.hpp"
00016
00017 #include <cstddef>
00018 #include <vector>
00019
00020 namespace Term
```

```
00021 {
00022
00028 //
00029 // @note the characters are represented by char32_t, representing their UTF-32 code point.
00032 class Window
00033 {
00034 public:
00035   Window(const std::size_t& columns, const std::size_t& rows);
00036
00037   std::size_t get_w() const;
00038
00039   std::size_t get_h() const;
00040
00041   void set_char(const std::size_t& column, const std::size_t& row, const char32_t& character);
00042
00043   void set_fg_reset(const std::size_t& column, const std::size_t& row);
00044
00045   void set_bg_reset(const std::size_t& column, const std::size_t& row);
00046
00047   void set_fg(const std::size_t& column, const std::size_t& row, const Color& color);
00048
00049   void set_bg(const std::size_t& column, const std::size_t& row, const Color& color);
00050
00051   void set_style(const std::size_t& column, const std::size_t& row, const Style& style);
00052
00053   void set_cursor_pos(const std::size_t& column, const std::size_t& row);
00054
00055   void set_h(const std::size_t&);
00056
00057   void print_str(const std::size_t& column, const std::size_t&, const std::string&, const std::size_t&
      = 0, bool = false);
00058
00059   void fill_fg(const std::size_t& column, const std::size_t&, const std::size_t&, const std::size_t&,
      const Color&);
00060
00061   void fill_bg(const std::size_t& column, const std::size_t&, const std::size_t&, const std::size_t&,
      const Color&);
00062
00063   void fill_style(const std::size_t& column, const std::size_t&, const std::size_t&, const
      std::size_t&, const Style&);
00064
00065   void print_border();
00066
00067   void print_rect(const std::size_t& column, const std::size_t&, const std::size_t&, const
      std::size_t&);
00068
00069   void clear();
00070
00071   bool insideWindow(const std::size_t& column, const std::size_t& row) const;
00072
00073   // TODO: add Window/Screen parameter here, to be used like this:
00074   // old_scr = scr;
00075   // scr.print_str(...)
00076   // scr.render(1, 1, old_scr)
00077   std::string render(const std::size_t&, const std::size_t&, bool);
00078
00079 private:
00080   std::size_t              index(const std::size_t& column, const std::size_t& row) const;
00081   Term::Screen             m_window{0, 0};
00082   Term::Cursor             m_cursor{1, 1};
00083   std::vector<char32_t>    m_chars;  // the characters in row first order
00084   std::vector<Term::Color> m_fg;
00085   std::vector<Term::Color> m_bg;
00086   std::vector<bool>        m_fg_reset;
00087   std::vector<bool>        m_bg_reset;
00088   std::vector<Style>       m_style;
00089
00090   char32_t get_char(const std::size_t& column, const std::size_t& row);
00091
00092   bool        get_fg_reset(const std::size_t& column, const std::size_t& row);
00093   bool        get_bg_reset(const std::size_t& column, const std::size_t& row);
00094   Term::Color get_fg(const std::size_t& column, const std::size_t& row);
00095   Term::Color get_bg(const std::size_t& column, const std::size_t& row);
00096   Term::Style get_style(const std::size_t& column, const std::size_t& row);
00097 };
00098
00099 }  // namespace Term
```

9.143 LICENSE File Reference

9.144 LICENSE

Go to the documentation of this file.

```
00001 Copyright (c) 2019 Ondřej Čertík
00002
00003 Permission is hereby granted, free of charge, to any person obtaining a copy
00004 of this software and associated documentation files (the "Software"), to deal
00005 in the Software without restriction, including without limitation the rights
00006 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00007 copies of the Software, and to permit persons to whom the Software is
00008 furnished to do so, subject to the following conditions:
00009
00010 The above copyright notice and this permission notice shall be included in
00011 all copies or substantial portions of the Software.
00012
00013 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00014 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00015 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00016 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00017 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00018 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
00019 THE SOFTWARE.
00020
00021 ------------------------------------------------------------------------------
00022
00023 The utf8_decode_step() function and the two defines UTF8_ACCEPT and UTF8_REJECT
00024 were taken from the [utf-8-misc](https://github.com/hoehrmann/utf-8-misc)
00025 project and are licensed as follows:
00026
00027 Copyright (c) 2014 Taylor R Campbell
00028 All rights reserved.
00029
00030 Redistribution and use in source and binary forms, with or without
00031 modification, are permitted provided that the following conditions
00032 are met:
00033 1. Redistributions of source code must retain the above copyright
00034   notice, this list of conditions and the following disclaimer.
00035 2. Redistributions in binary form must reproduce the above copyright
00036   notice, this list of conditions and the following disclaimer in the
00037   documentation and/or other materials provided with the distribution.
00038
00039 THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS''
00040 AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00041 IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00042 ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE
00043 LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00044 CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00045 SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00046 INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00047 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00048 ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00049 POSSIBILITY OF SUCH DAMAGE.
00050
00051 ------------------------------------------------------------------------------
00052
00053 The enumeration classes `style`, `fg`, `bg`, `fgB` and `bgB` were taken from
00054 the [rang](https://github.com/agauniyal/rang) project, which is licensed as:
00055
00056 This is free and unencumbered software released into the public domain.
00057
00058 Anyone is free to copy, modify, publish, use, compile, sell, or
00059 distribute this software, either in source code form or as a compiled
00060 binary, for any purpose, commercial or non-commercial, and by any
00061 means.
00062
00063 In jurisdictions that recognize copyright laws, the author or authors
00064 of this software dedicate any and all copyright interest in the
00065 software to the public domain. We make this dedication for the benefit
00066 of the public at large and to the detriment of our heirs and
00067 successors. We intend this dedication to be an overt act of
00068 relinquishment in perpetuity of all present and future rights to this
00069 software under copyright law.
00070
00071 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00072 EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00073 MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00074 IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00075 OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00076 ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
00077 OTHER DEALINGS IN THE SOFTWARE.
00078
00079 For more information, please refer to <http://unlicense.org>
00080
00081 ------------------------------------------------------------------------------
00082
00083 The file examples/kilo.cpp was taken from the
00084 [kilo-src](https://github.com/snaptoken/kilo-src) project, which is licensed
00085 as:
00086
00087 Copyright (c) 2016, Salvatore Sanfilippo <antirez at gmail dot com>
```

```
00088
00089 All rights reserved.
00090
00091 Redistribution and use in source and binary forms, with or without
00092 modification, are permitted provided that the following conditions are met:
00093
00094 * Redistributions of source code must retain the above copyright notice,
00095   this list of conditions and the following disclaimer.
00096
00097 * Redistributions in binary form must reproduce the above copyright notice,
00098   this list of conditions and the following disclaimer in the documentation
00099   and/or other materials provided with the distribution.
00100
00101 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00102 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00103 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00104 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
00105 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00106 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00107 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00108 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00109 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00110 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```