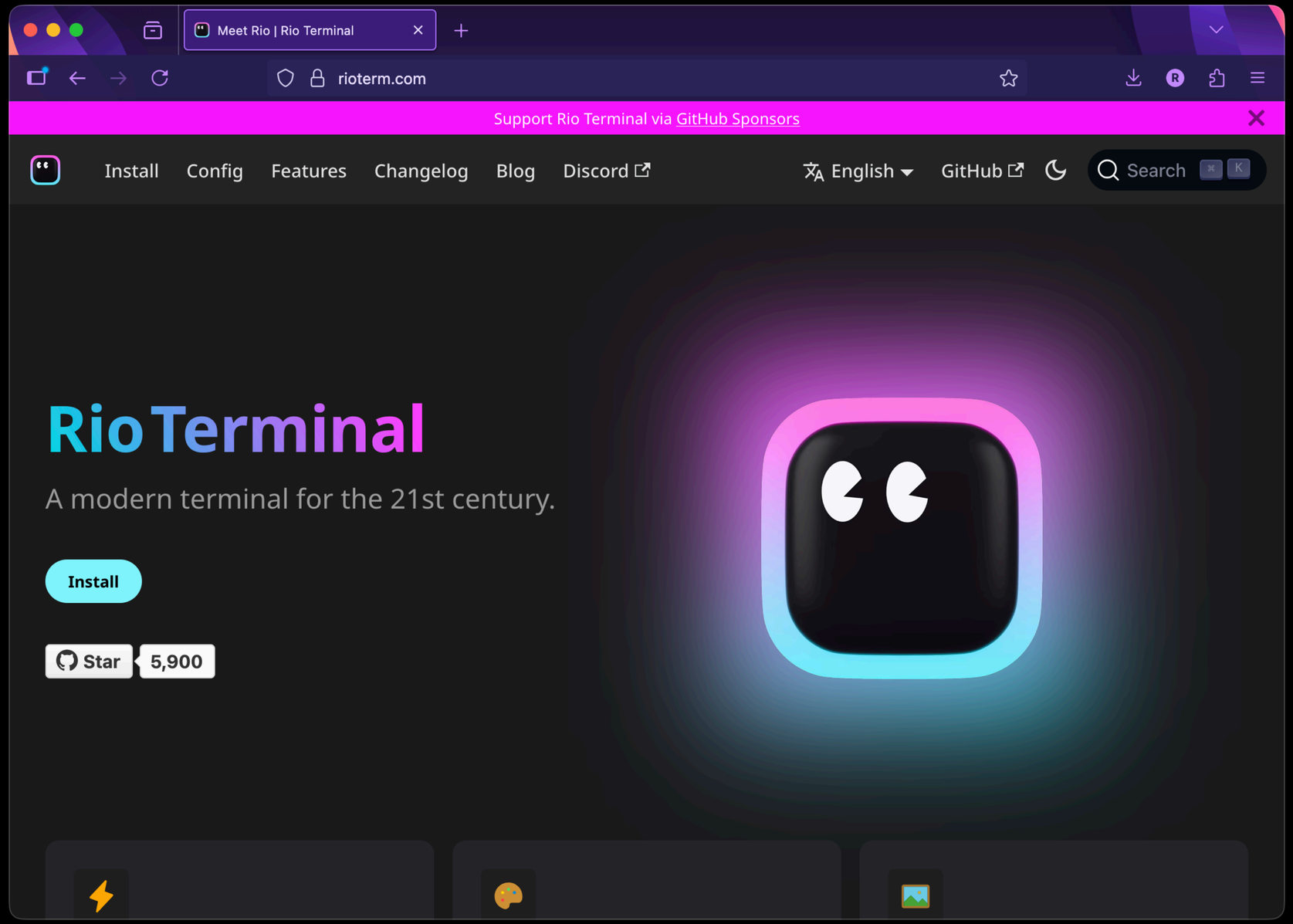
Terminals Are Just Text? Think Again

Raphael Amorim

github.com/raphamorim


Meet Rio | Rio Terminal
rioterm.com
Support Rio Terminal via GitHub Sponsors
Install
Config
Features
Changelog
Blog
Discord
English
GitHub
Search
Rio Terminal
A modern terminal for the 21st century.
Install
Star
5,900

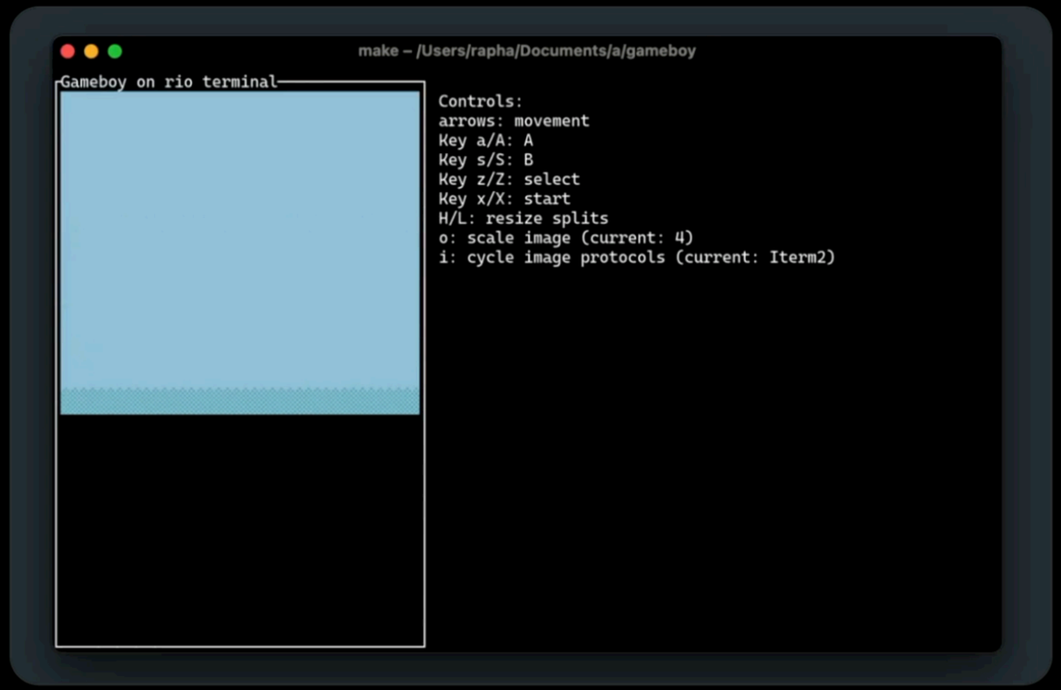
Pinned

Raphael Amorim
@raphamorims

Now you can play GAMEBOY GAMES ON TERMINAL. Any good terminal that can run Sixel, iTerm2 or Kitty image protocol.

(P.S: I wrote this at Christmas time, so still have to fix audio and normalize keyup)

github.com/raphamorim/gam...



3:06 pm · 30 Dec 2024 · **130.2K** Views

View post engagements

34 254 2K 1K

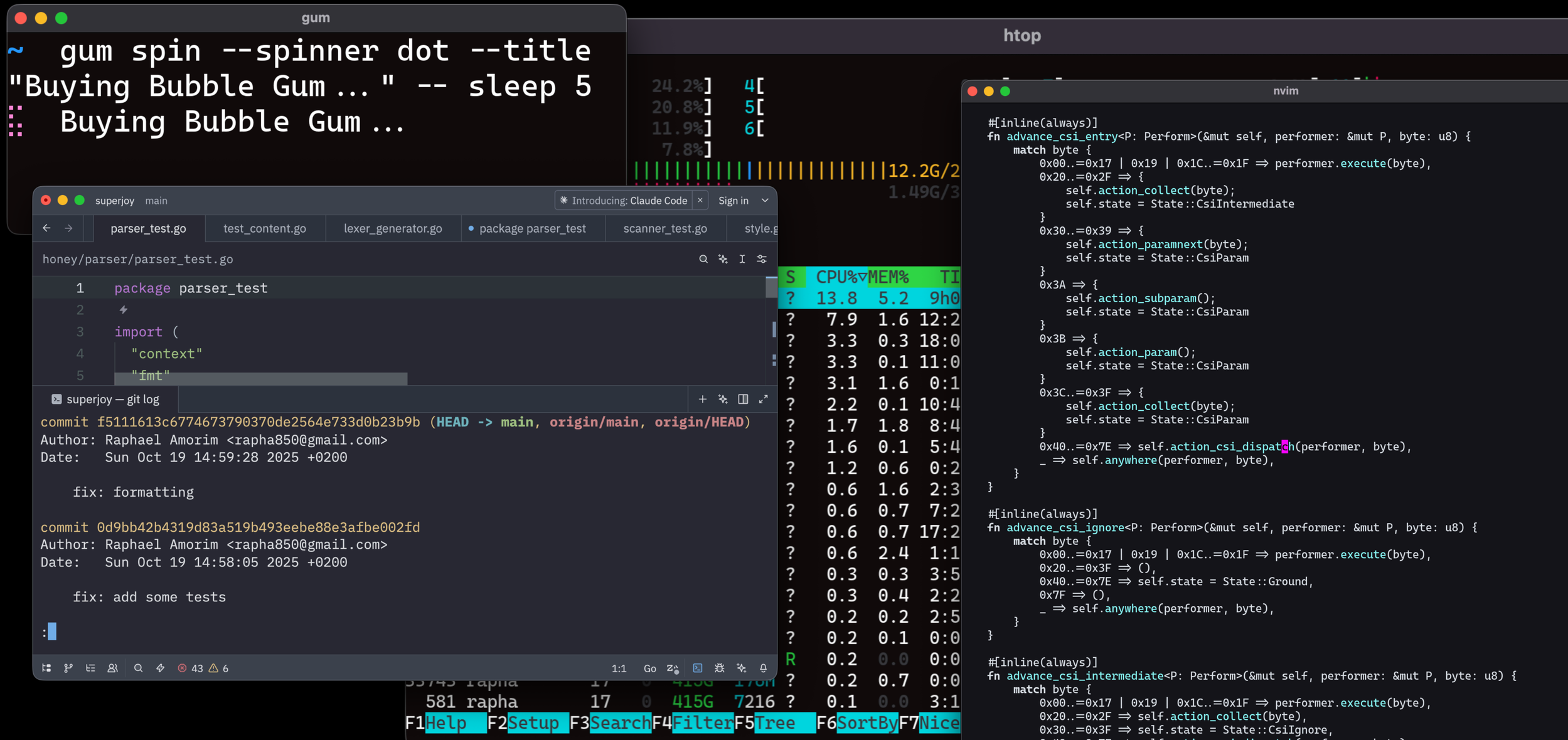
I ❤️ Terminals

Disclaimer

This talk will focus on Unix-based operating systems. I wanted to cover others as well, but due to time constraints, that would have meant overlooking the Unix side.

Ok, let’s dive in

Most of us use terminal every day.



In conventional ways and

unconventional ways!

```
    Update README

commit a726b515db1540592082314dee701d8dc7493e11
Author: Hayaki Saito <user@zuse.jp>
Date:   Thu Aug 14 12:44:55 2014 +0900

    Introduce -dropframe, -ignoredelay options

commit 43c9f7a3386a3655dc98cd49e0366e5eea83ba8f
Author: Hayaki Saito <user@zuse.jp>
Date:   Wed Aug 13 17:12:36 2014 +0900

    Introduce new options: -fixedpal, -diffuse, -scene-threshold

commit 76b5463a9fb220d414617677a5260de8c69da460
Author: Hayaki Saito <user@zuse.jp>
Date:   Wed Aug 13 00:14:12 2014 +0900

    Update README

commit b7a1b5b7efcac561af7e743d3bdabbf56cac1fcb
Author: Hayaki Saito <user@zuse.jp>
Date:   Tue Aug 12 23:59:10 2014 +0900

    Add some options: -reqcolors, -row, -col

commit 95f5e0b27e8e5029b6746e978bc7766862a8eb65
Author: Hayaki Saito <user@zuse.jp>
Date:   Thu Aug 7 17:23:28 2014 +0900

    Use av_calloc instead of the pair of malloc/memset

VPS FFmpeg-SIXEL% (sixel) time ./ffmpeg -i 'https://www.youtube.com/watch?v=ixaMZPPmVG0' -pix_fmt rg
b24 -f sixel -loglevel quiet -s 800x450 -
bck-i-search: ./_
```


mlterm
Administrator
インターネット
Internet Explorer
電子メール
Outlook Express
Windows XP ツアー
Windows Media Player
Windows Messenger
MSN
ファイルと設定の転送ウィザード
すべてのプログラム(P)
マイ ドキュメント
最近使ったファイル(D)
マイ ピクチャ
マイ ミュージック
マイ コンピュータ
コントロール パネル(C)
プログラムのアクセスと既定の設定
プリンタと FAX
ヘルプとサポート(H)
検索(S)
ファイル名を指定して実行(R)...
ごみ箱
表示できない項目がいくつかあります
[スタート] メニューに追加したすべての項目を表示する領域がありません。表示するには、小さいメニュー アイコンを選択したり、項目数を減らしたりしてください。
15:33
events: 2558 button: [0] cursor: (77, 300)

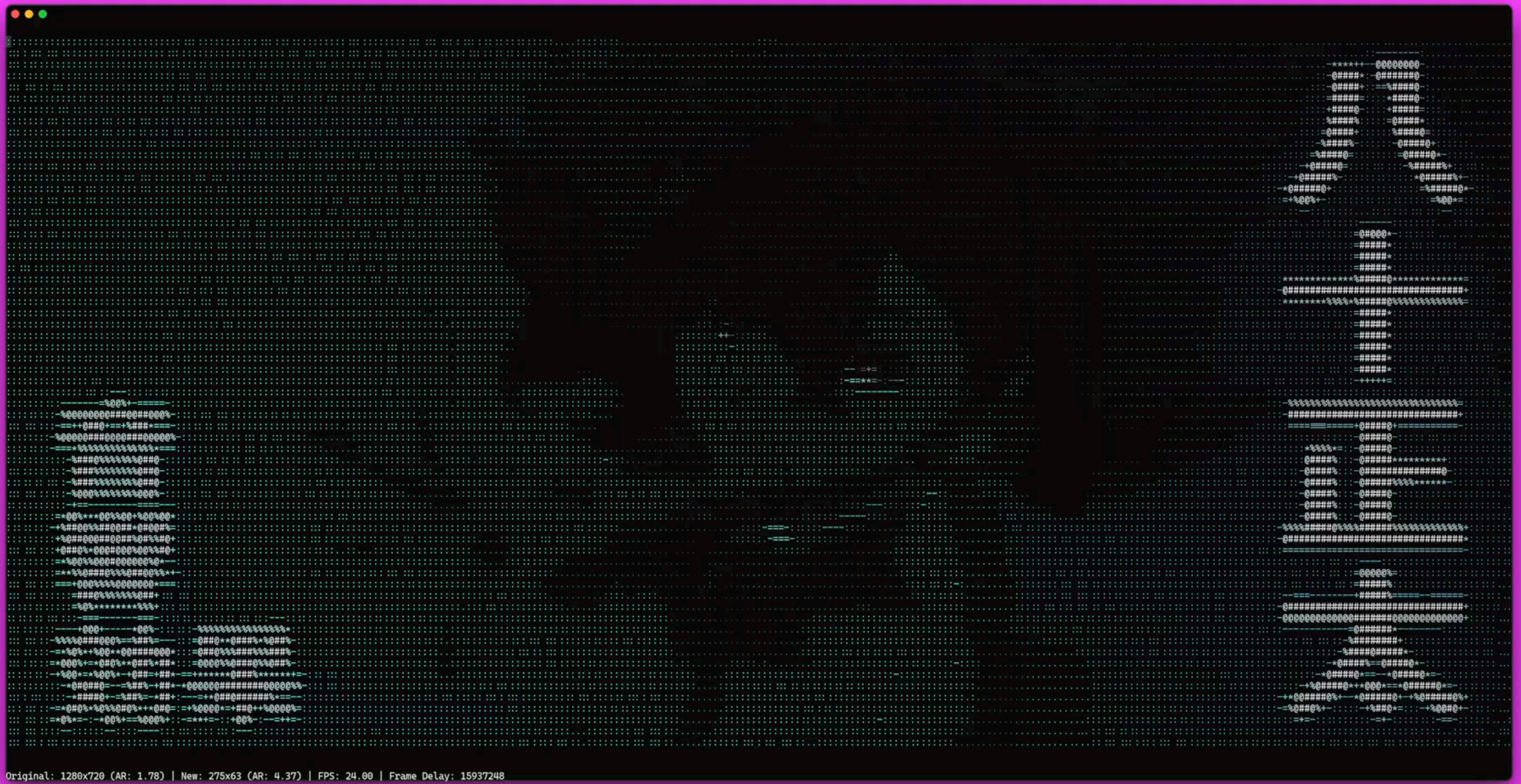
zig-out/bin/terminal ~/p/terminal-doom


```
…/terminal-doom  main  ⚡ echo $TERM_PROGRAM_VERSION
0.1.0-main+4f8a7d32

…/terminal-doom  main  ⚡ ghostty +version
Ghostty 0.1.0-main+4f8a7d32

Build Config
  - Zig version: 0.13.0
  - build mode : builtin.OptimizeMode.ReleaseFast
  - app runtime: apprt.Runtime.none
  - font engine: font.main.Backend.coretext
  - renderer   : renderer.Metal
  - libxev     : main.Backend.kqueue

…/terminal-doom  main  ⚡
```

```
55  pub export fn DG_DrawFrame() callconv(.C) void {
56      // We need to have a window size before continuing
57      const win = state.loop.vaxis.window();
58      if (win.screen.width == 0) {
59          while (state.loop.tryEvent()) |event| {
60              switch (event) {
61                  .winsize => |ws| state.loop.vaxis.resize(std.heap.c_alloca
62                  else => {},
63              }
64          }
65          return;
66      }
67      translateDoomBufferToRGBA();
68
69      var pixels = zigimg.Image{
70          .width = 640,
71          .height = 400,
72          .pixels = zigimg.color.PixelStorage.initRawPixels(&DG_ScreenBuffer
73      };
NOR 18% main src/main.zig                                   1 sel  67:1
```

~

```
config.sub              Makefile.am
configure               Makefile.in
configure.ac            missing
converters              NEWS
depcomp                 package.json
examples                package.json.in
gnuplot                 package.json.in.in
images                  perl
img2sixel.exe.core      php
include                 py-compile
install-sh              python
libsixel.pc             README.md
libsixel.pc.in          ruby
libtool                 src
LICENSE                 stamp-h1
LICENSE.images          tools
LICENSE.pngsuite        wic
(reverse-i-search)`': 
```



Original: 1280x720 (AR: 1.78) | New: 275x63 (AR: 4.37) | FPS: 24.00 | Frame Delay: 15937248

Terminals are used in many different use cases.

But how it actually works?

Teletypes

* Tom Hanks is not really connected to our talk, I just thought he has a nice collection

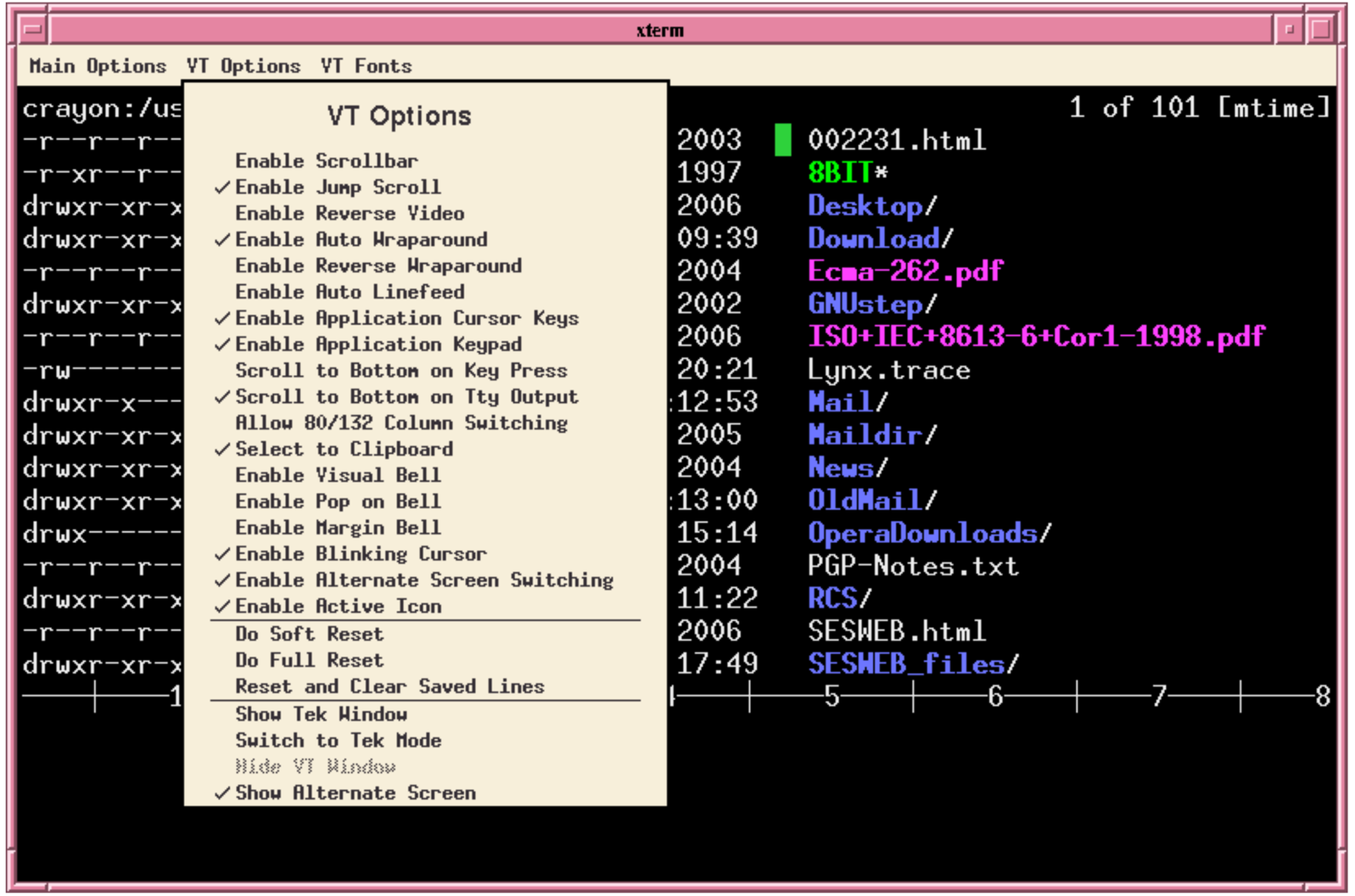
Teletype Writers (TTYs)

```
zsh
~  tty
/dev/ttys003
~
```

DEC VT05

VT100


SLMIN .SYS 12P 20-Dec-85
XL .SYS 4P 20-Dec-85
SP .SYS 6P 13-Aug-86
RT11SJ.SYS 78P 13-Aug-86
NL .SYS 2P 13-Aug-86
SQ .SYS 5P 31-May-85
BASIC .SAV 56 24-May-79
DATIME.SAV 4 20-Dec-85
DUMP .SAV 9 20-Dec-85
TSXMOD.SAV 78 27-Nov-92
HARRIS.SAV 41 12-Jun-85
START .P36 2 21-Dec-91
EM1A .STM 19P 02-Feb-93
STAND .LIN 12P 15-Aug-83
EM1B .STM 19 11-Feb-93
JKFLIP.FOR 3 08-Mar-96
ANNOT .SAV 38 18-Apr-93
DUO .DIR 18 16-Jul-96
DIR .DIF 26 16-Jul-96
DEMOBG.OBJ 1 -BAD-
114 Files, 5949 Blocks
14433 Free blocks
.o
VM .SYS 3P 13-Aug-86
LD .SYS 8P 23-Aug-86
DL .SYS 5P 13-Aug-86
DU .SYS 6P 13-Aug-86
TT .SYS 2P 13-Aug-86
RL02DC.SYS 71P 21-Nov-84
BINCON.SAV 24 20-Dec-85
DIR .SAV 19 20-Dec-85
DUP .SAV 47 20-Dec-85
FORTRA.SAV 206 21-May-85
LET .SAV 5 20-Dec-85
RETRO .OBJ 1519P 16-May-88
TSXUCL.TSX 22 07-Mar-92
TSXV6 .MSG 1P 04-Sep-95
JKFLIP.SAV 30 08-Mar-96
RT11FB.SYS 93P 20-Dec-85
TSXP23.NEW 1200P 27-Nov-92
DL .DIR 7 16-Jul-96
DEMOFG.OBJ 1 -BAD-
EVAN .ID 1
BC-BS 20768
digital VT100



XTerm

Created in 1984.

It started based on the DEC VT102 terminal specs and later incorporated features from other DEC terminals like the VT220, VT320, VT420, VT520, and Tektronix 4014.

PTY (Pseudo-Terminal)

In some operating systems, including Unix-like systems, a pseudoterminal, pseudotty, or PTY is a pair of pseudo-device endpoints (files) which establish an asynchronous, bidirectional communication (IPC) channel (with two ports) between two or more processes


Terminal Emulators
$
>_
>_
>_
$W
>_

Terminal Emulator <-> PTY <-> Process*

* Bash,Fish,Vim,etc…

User input →
terminal writes to Pty input buffer →
shell reads Pty input buffer and interprets them →
shell writes to Pty output buffer →
terminal interprets them →
renders

Program writes →
terminal reads Pty →
terminal parsers buffer →
(optionally) update terminal view/state →
renders.


zsh
~ tty
/dev/ttys008
~ cat /dev/ttys008
ok folks
how is it going
~ echo "ok folks" >> /dev/ttys008
~ echo "how is it going" >> /dev/ttys008
~

The buffer frequently contains ANSI escape codes.

ANSI escape sequences are a standard for in-band signaling to control cursor location, color, font styling, and other options.


zsh
~ cat /dev/ttys008
Gimme some bright red
~ echo "\x1b[91m
Gimme some bright
red\x1b[0m" >> /
dev/ttys008
~

ANSI sequences were introduced in the 1970s to replace vendor-specific sequences and became widespread in the computer equipment market by the early 1980s.

* Although hardware text terminals have become increasingly rare in the 21st century, the relevance of the ANSI standard persists because a great majority of terminal emulators and command consoles interpret at least a portion of the ANSI standard.

Certain sequences of bytes, most starting with an ASCII escape character and a bracket character, are embedded into text. The terminal interprets these sequences as commands, rather than text to display verbatim.

Popular C0 control codes (not an exhaustive list)

^	C0	Abbr	C escape sequence	Name	Effect
^G	0x07	BEL	`\a`	Bell	Makes an audible noise.
^H	0x08	BS	`\b`	Backspace	Moves the cursor left (but may "backwards wrap" if cursor is at start of line).
^I	0x09	HT	`\t`	Tab	Moves the cursor right to next tab stop.
^J	0x0A	LF	`\n`	Line Feed	Moves to next line, scrolls the display up if at bottom of the screen. Usually does not move horizontally, though programs should not rely on this.
^L	0x0C	FF	`\f`	Form Feed	Move a printer to top of next page. Usually does not move horizontally, though programs should not rely on this. Effect on video terminals varies.
^M	0x0D	CR	`\r`	Carriage Return	Moves the cursor to column zero.
^[	0x1B	ESC	`\x1B`, `\033`	Escape	Starts all the escape sequences

Some type Fe (C1 set element) ANSI escape sequences (not an exhaustive list)

Code	C1	Abbr	Name	Effect
ESC N	0x8E	SS2	Single Shift Two	Select a single character from one of the alternative character sets. SS2 selects the G2 character set, and SS3 selects the G3 character set.[17] In a 7-bit environment, this is followed by one or more GL bytes (0x20–0x7F) specifying a character from that set.[15]:9.4 In an 8-bit environment, these may instead be GR bytes (0xA0–0xFF).[15]:8.4
ESC O	0x8F	SS3	Single Shift Three	
ESC P	0x90	DCS	Device Control String	Terminated by ST.[16]:5.6 Xterm's uses of this sequence include defining User-Defined Keys, and requesting or setting Termcap/Terminfo data.[17]
ESC [	0x9B	CSI	Control Sequence Introducer	Starts most of the useful sequences, terminated by a byte in the range 0x40 through 0x7E.[16]:5.4
ESC \	0x9C	ST	String Terminator	Terminates strings in other controls.[16]:8.3.143
ESC]	0x9D	OSC	Operating System Command	Starts a control string for the operating system to use, terminated by ST.[16]:8.3.89
ESC X	0x98	SOS	Start of String	Takes an argument of a string of text, terminated by ST.[16]:5.6 The uses for these string control sequences are defined by the application[16]:8.3.2,8.3.128 or privacy discipline.[16]:8.3.94 These functions are rarely implemented and the arguments are ignored by xterm.[17] Some Kermit clients allow the server to automatically execute Kermit commands on the client by embedding them in APC sequences; this is a potential security risk if the server is untrusted.[18]
ESC ^	0x9E	PM	Privacy Message	
ESC _	0x9F	APC	Application Program Command	

Some ANSI control sequences (not an exhaustive list)

Code[a]	Abbr	Name	Effect
`CSI n A`	CUU	Cursor Up	Moves the cursor *n* (default `1`) cells in the given direction. If the cursor is already at the edge of the screen, this has no effect.
`CSI n B`	CUD	Cursor Down	
`CSI n C`	CUF	Cursor Forward	
`CSI n D`	CUB	Cursor Back	
`CSI n E`	CNL	Cursor Next Line	Moves cursor to beginning of the line *n* (default `1`) lines down. (not ANSI.SYS)
`CSI n F`	CPL	Cursor Previous Line	Moves cursor to beginning of the line *n* (default `1`) lines up. (not ANSI.SYS)
`CSI n G`	CHA	Cursor Horizontal Absolute	Moves the cursor to column *n* (default `1`). (not ANSI.SYS)
`CSI n ; m H`	CUP	Cursor Position	Moves the cursor to row *n*, column *m*. The values are 1-based, and default to `1` (top left corner) if omitted. A sequence such as `CSI ;5H` is a synonym for `CSI 1;5H` as well as `CSI 17;H` is the same as `CSI 17H` and `CSI 17;1H`
`CSI n J`	ED	Erase in Display	Clears part of the screen. If *n* is `0` (or missing), clear from cursor to end of screen. If *n* is `1`, clear from cursor to beginning of the screen. If *n* is `2`, clear entire screen (and moves cursor to upper left on DOS ANSI.SYS). If *n* is `3`, clear entire screen and delete all lines saved in the scrollback buffer (this feature was added for xterm and is supported by other terminal applications).
`CSI n K`	EL	Erase in Line	Erases part of the line. If *n* is `0` (or missing), clear from cursor to the end of the line. If *n* is `1`, clear from cursor to beginning of the line. If *n* is `2`, clear entire line. Cursor position does not change.
`CSI n S`	SU	Scroll Up	Scroll whole page up by *n* (default `1`) lines. New lines are added at the bottom. (not ANSI.SYS)
`CSI n T`	SD	Scroll Down	Scroll whole page down by *n* (default `1`) lines. New lines are added at the top.

Some popular private sequences

Code	Abbr	Name	Effect
`CSI s`	`SCP`, `SCOSC`	Save Current Cursor Position	Saves the cursor position/state in SCO console mode.[19] In vertical split screen mode, instead used to set (as `CSI n ; n s`) or reset left and right margins.[20]
`CSI u`	`RCP`, `SCORC`	Restore Saved Cursor Position	Restores the cursor position/state in SCO console mode.[21]
`CSI ? 25 h`	`DECTCEM`		Shows the cursor, from the VT220.
`CSI ? 25 l`	`DECTCEM`		Hides the cursor.
`CSI ? 1004 h`			Enable reporting focus. Reports whenever terminal emulator enters or exits focus as `ESC [I` and `ESC [O`, respectively.
`CSI ? 1004 l`			Disable reporting focus.
`CSI ? 1049 h`			Enable alternative screen buffer, from xterm
`CSI ? 1049 l`			Disable alternative screen buffer, from xterm
`CSI ? 2004 h`			Turn on bracketed paste mode.[22] In bracketed paste mode, text pasted into the terminal will be surrounded by `ESC [200~` and `ESC [201~`; programs running in the terminal should not treat characters bracketed by those sequences as commands (Vim, for example, does not treat them as commands).[23] From xterm[24]
`CSI ? 2004 l`			Turn off bracketed paste mode.

CSI *n* m (Select Graphic Rendition parameters)

n	Name	Note
0	Reset *or* normal	All attributes become turned off
1	Bold or increased intensity	As with faint, the color change is a PC (SCO / CGA) invention.[26][*better source needed*]
2	Faint, decreased intensity, *or* dim	May be implemented as a light font weight like bold.[27]
3	Italic	Not widely supported. Sometimes treated as inverse or blink.[26]
4	Underline	Style extensions exist for Kitty, VTE, mintty, iTerm2 and Konsole.[28][29][30]
5	Slow blink	Sets blinking to less than 150 times per minute
6	Rapid blink	MS-DOS ANSI.SYS, 150+ per minute; not widely supported
7	Reverse video *or* invert	Swap foreground and background colors; inconsistent emulation[31][*dubious – discuss*]
8	Conceal *or* hide	Not widely supported.
9	Crossed-out, *or* strike	Characters legible but marked as if for deletion. Not supported in Terminal.app.
10	Primary (default) font	
11–19	Alternative font	Select alternative font $n - 10$
20	Fraktur (Gothic)	Rarely supported
21	Doubly underlined; or: not bold	Double-underline per ECMA-48,[16]:8.3.117 but instead disables bold intensity on several terminals, including in the Linux kernel's console before version 4.17.[32]

Of course you have terminal protocols extensions.

Extensions started from DEC, which provided backwards-compatible series, extending from the first ANSI-compatible model introduced in 1978 (the VT100)

Sixel,
iTerm2 (inline) Image Protocol,
Kitty Keyboard Protocol,
Kitty Image Protocol,
 …

zsh

```
~  cat /dev/ttys008
```

```
~  img2sixel /Users/rapha/Downloads
/tiger.png >> /dev/ttys008
~
```

There’s actually much more stuff but for now let’s focus on what we just learned.

Program writes \033[2J

CSI 2J: Erase entire screen

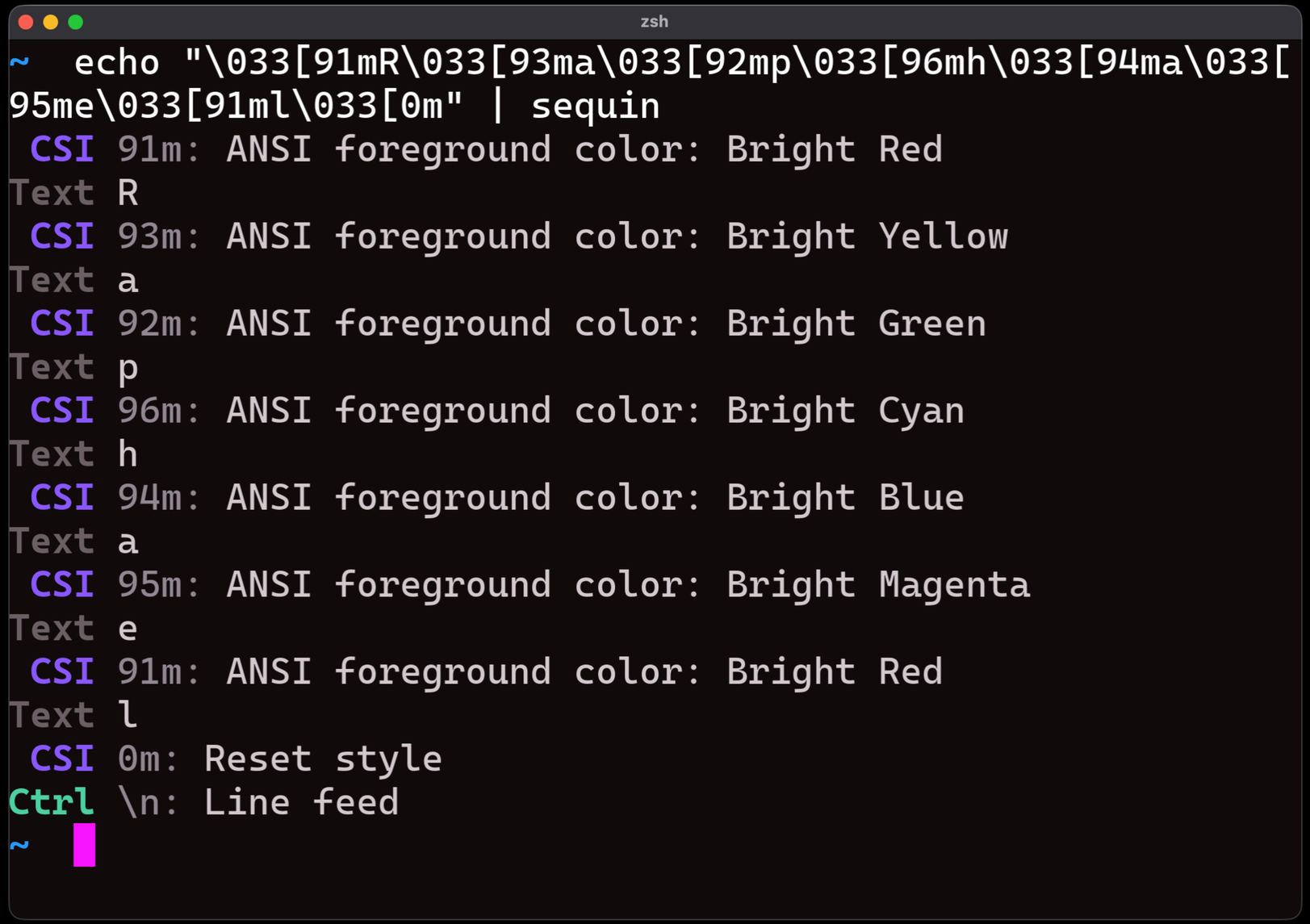
Program writes \033[2J

Program writes "hello peeps!"

CSI 2J Erase entire screen

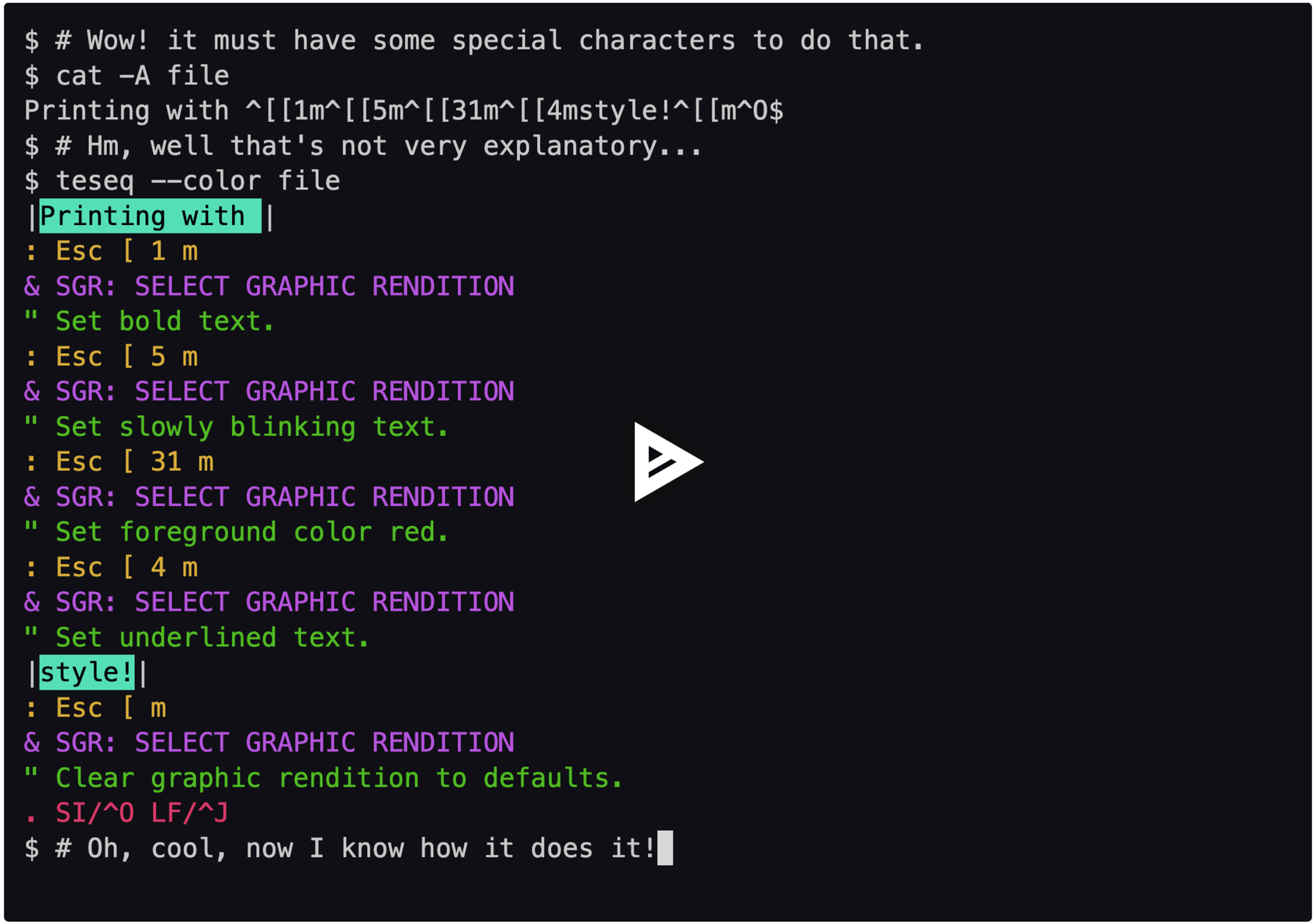
Text   Hello peeps!

Parsing data



charmbracelet/sequin

GNU Teseq



```rust
//! Parse input from stdin and log actions on stdout
use std::io::{self, Read};

use vte::{Params, Parser, Perform};

/// A type implementing Perform that just logs actions
struct Log;

impl Perform for Log {
    fn print(&mut self, c: char) {
        println!("[print] {:?}", c);
    }

    fn execute(&mut self, byte: u8) {
        println!("[execute] {:02x}", byte);
    }

    fn hook(&mut self, params: &Params, intermediates: &[u8], ignore: bool, c: char) {
        println!(
            "[hook] params={:?}, intermediates={:?}, ignore={:?}, char={:?}",
            params, intermediates, ignore, c
        );
    }

    fn put(&mut self, byte: u8) {
        println!("[put] {:02x}", byte);
    }

    fn unhook(&mut self) {
        println!("[unhook]");
    }

    fn osc_dispatch(&mut self, params: &[&[u8]], bell_terminated: bool) {
        println!("[osc_dispatch] params={:?} bell_terminated={}", params, bell_terminated);
    }

    fn csi_dispatch(&mut self, params: &Params, intermediates: &[u8], ignore: bool, c: char) {
        println!(
            "[csi_dispatch] params={:#?}, intermediates={:?}, ignore={:?}, char={:?}",
            params, intermediates, ignore, c
        );
    }

    fn esc_dispatch(&mut self, intermediates: &[u8], ignore: bool, byte: u8) {
        println!(
            "[esc_dispatch] intermediates={:?}, ignore={:?}, byte={:02x}",
            intermediates, ignore, byte
        );
    }
}
```

```rust
fn main() {
    let input = io::stdin();
    let mut handle = input.lock();

    let mut statemachine = Parser::new();
    let mut performer = Log;

    let mut buf = [0; 2048];

    loop {
        match handle.read(&mut buf) {
            Ok(0) => break,
            Ok(n) => statemachine.advance(&mut performer, &buf[..n]),
            Err(err) => {
                println!("err: {}", err);
                break;
            },
        }
    }
}
```

Pty

forkpty

Name

forkpty -- Create a new process attached to an available pseudo-terminal

Synopsis

```
#include <pty.h>

int forkpty(int * amaster, char * name, struct termios * termp, struct winsize * w
```

Description

The `forkpty()` function shall find and open a pseudo-terminal device pair in the same manner as the pseudo-terminal is available, `forkpty()` shall create a new process in the same manner as the `fork(` new process for login in the same manner as `login_tty()`.

If *termp* is not null, it shall refer to a `termios` structure that shall be used to initialize the characterist is not null, it shall refer to a `winsize` structure used to initialize the window size of the slave device.

Return Value

On success, the parent process shall return the process id of the child, and the child shall return 0. Or created, -1 shall be returned, and `errno` shall be set appropriately. On success, the parent process sha the master side of the pseudo-terminal in the location referenced by *amaster*, and, if *name* is not NUl device in *name*.

Errors

EAGAIN

Unable to create a new process.

ENOENT

There are no available pseudo-terminals.

ENOMEM

Insufficient memory was available.

openpty(3) — Linux manual page

openpty(3) **Library Functions Manual** **openpty(3)**

NAME top

openpty, login_tty, forkpty - terminal utility functions

LIBRARY top

System utilities library (*libutil*, *-lutil*)

SYNOPSIS top

```
#include <pty.h>

int openpty(int *amaster, int *aslave, char *name,
            const struct termios *termp,
            const struct winsize *winp);
pid_t forkpty(int *amaster, char *name,
            const struct termios *termp,
            const struct winsize *winp);

#include <utmp.h>

int login_tty(int fd);
```

DESCRIPTION top

The **openpty**() function finds an available pseudoterminal and returns file descriptors for the master and slave in *amaster* and *aslave*. If *name* is not NULL, the filename of the slave is returned in *name*. If *termp* is not NULL, the terminal parameters of the slave will be set to the values in *termp*. If *winp* is not NULL, the window size of the slave will be set to the values in *winp*.

The **login_tty**() function prepares for a login on the terminal referred to by the file descriptor *fd* (which may be a real terminal device, or the slave of a pseudoterminal as returned by **openpty**()) by creating a new session, making *fd* the controlling terminal for the calling process, setting *fd* to be the standard input, output, and error streams of the current process, and closing *fd*.

The **forkpty**() function combines **openpty**(), fork(2), and **login_tty**() to create a new process operating in a pseudoterminal. A file descriptor referring to master side of the pseudoterminal is returned in *amaster*. If *name* is not NULL, the buffer it points to is used to return the filename of the slave. The *termp* and *winp* arguments, if not NULL, will determine the terminal attributes and window size of the slave side of the pseudoterminal.

RETURN VALUE top

posix_openpt(3) — Linux manual page

posix_openpt(3) **Library Functions Manual** **posix_openp**

NAME top

posix_openpt - open a pseudoterminal device

LIBRARY top

Standard C library (*libc*, *-lc*)

SYNOPSIS top

```
#include <stdlib.h>
#include <fcntl.h>

int posix_openpt(int flags);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

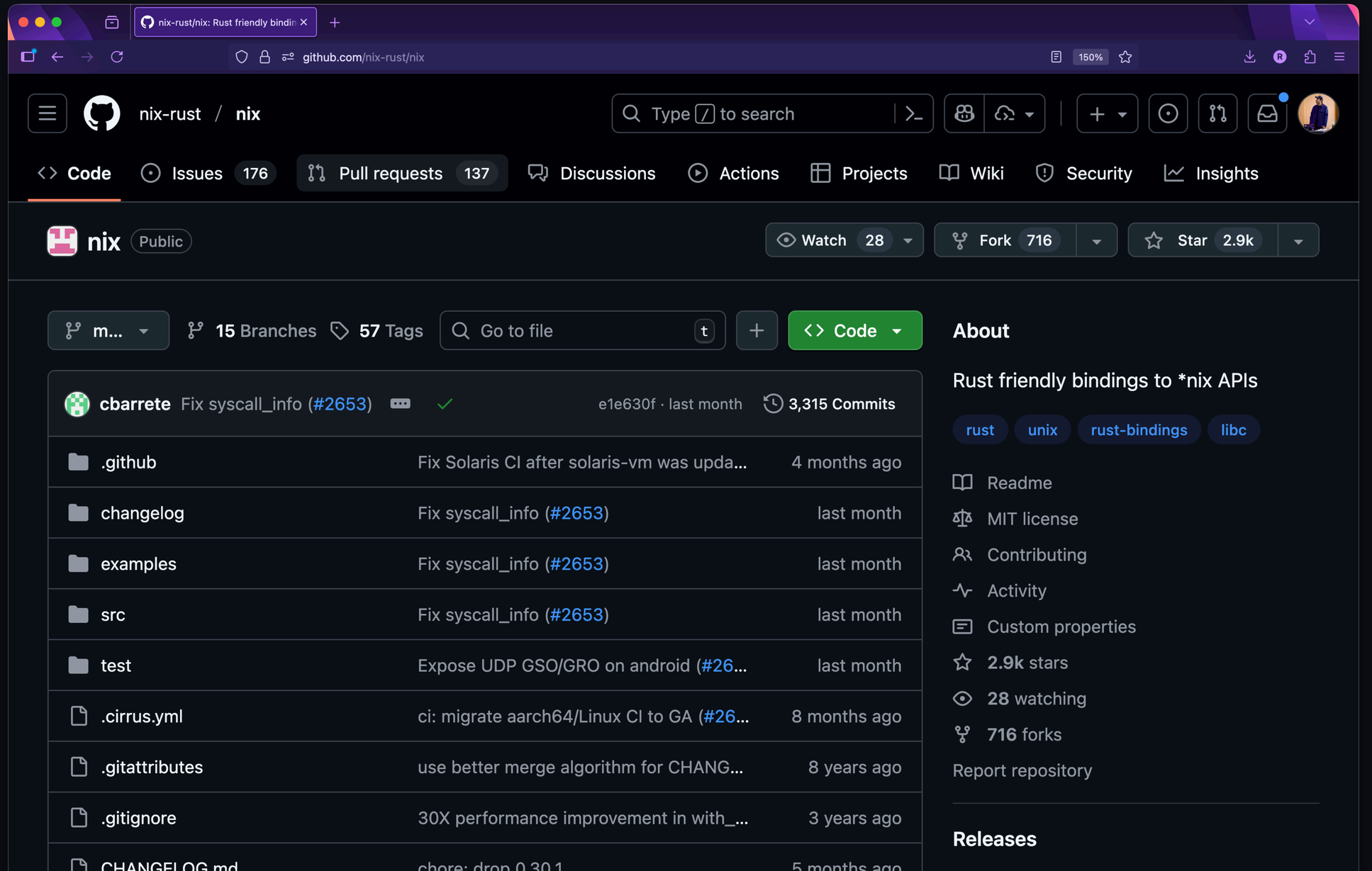
posix_openpt():
_XOPEN_SOURCE >= 600

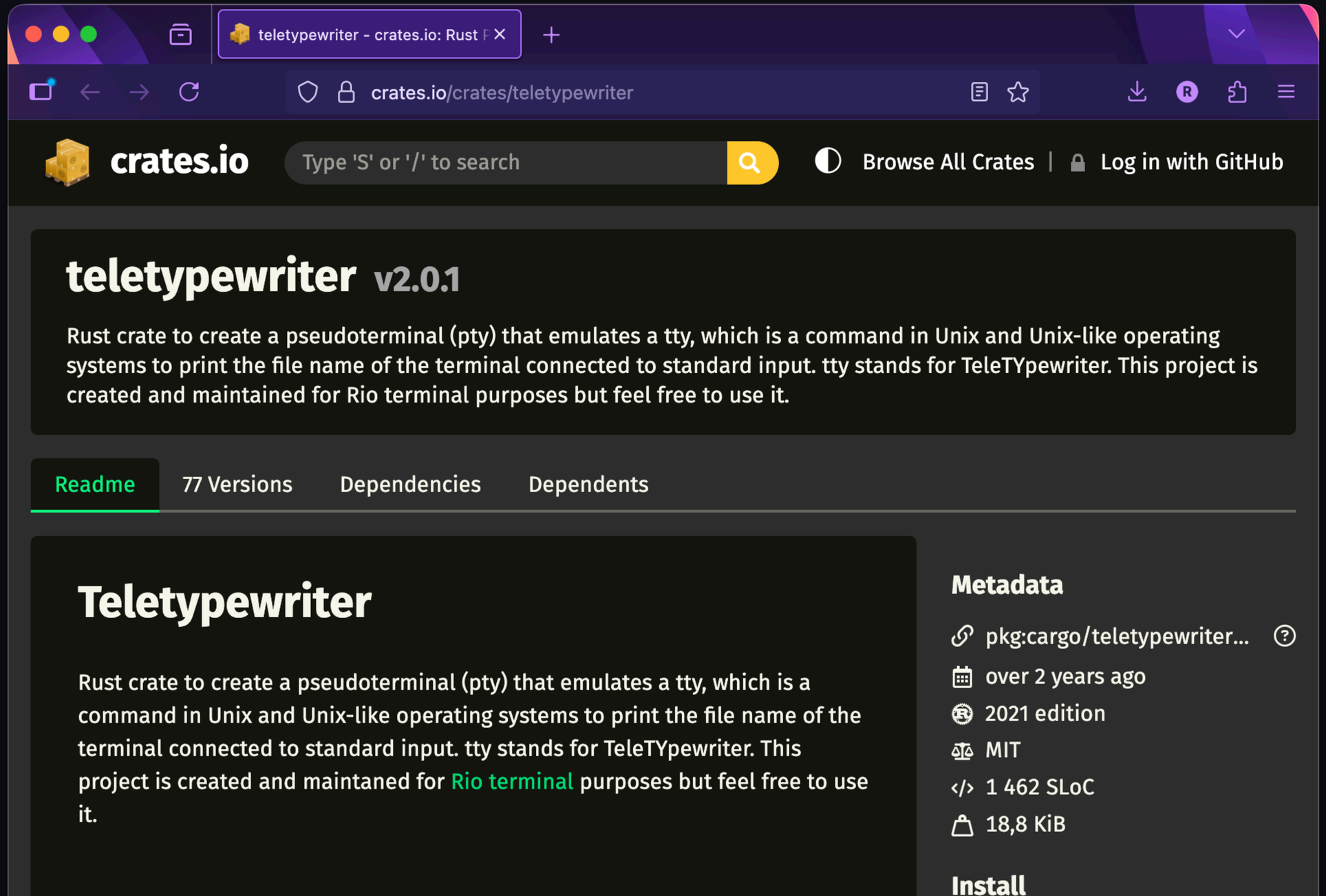
```rust
pub fn create_pty_with_spawn(

    let res = unsafe {
        openpty(
            &mut main as *mut _,
            &mut child as *mut _,
            ptr::null_mut(),
            &term as *const libc::termios,
            &winsize as *const _,
        )
    };

    if res < 0 {
        return Err(Error::other("openpty failed"));
    }

    let mut shell_program = shell;

    let user = match ShellUser::from_env() {
        Ok(data) => data,
        Err(..) => ShellUser {
            shell: shell.to_string(),
            ..Default::default()
        },
    };

    if shell.is_empty() {
        shell_program = &user.shell;
    }

    tracing::info!("spawn {:?} {:?}", shell_program, args);
```

rio / teletypewriter / src / unix / **mod.rs**

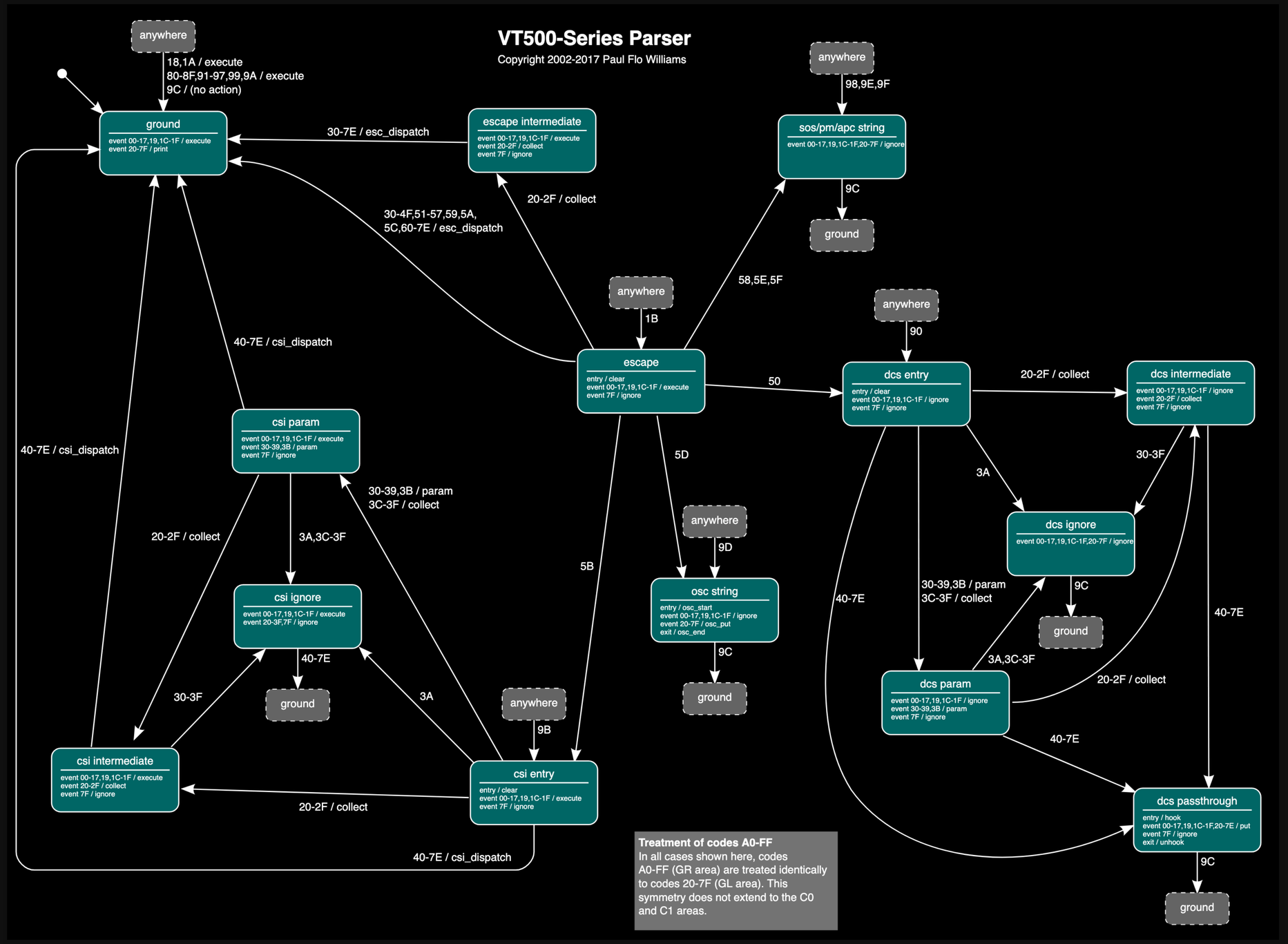
Code | Blame | 1001 lines (871 loc) · 29.1 KB

```rust
#[cfg(all(target_os = "linux", not(target_env = "musl")))]
const TIOCSWINSZ: libc::c_ulong = 0x5414;
#[cfg(all(target_os = "linux", target_env = "musl"))]
const TIOCSWINSZ: libc::c_int = 0x5414;
#[cfg(target_os = "freebsd")]
const TIOCSWINSZ: libc::c_ulong = 0x80087467;
#[cfg(target_os = "macos")]
const TIOCSWINSZ: libc::c_ulong = 2148037735;

#[link(name = "util")]
extern "C" {
    fn forkpty(
        main: *mut libc::c_int,
        name: *mut libc::c_char,
        termp: *const libc::termios,
        winsize: *const Winsize,
    ) -> libc::pid_t;

    fn openpty(
        main: *mut libc::c_int,
        child: *mut libc::c_int,
        name: *mut libc::c_char,
        termp: *const libc::termios,
        winsize: *const Winsize,
    ) -> libc::pid_t;

    fn waitpid(
```


nix-rust/nix: Rust friendly bindin
github.com/nix-rust/nix
150%
nix-rust / nix
Type / to search
Code
Issues 176
Pull requests 137
Discussions
Actions
Projects
Wiki
Security
Insights
nix
Public
Watch 28
Fork 716
Star 2.9k
m...
15 Branches
57 Tags
Go to file
t
Code
cbarrete Fix syscall_info (#2653)
e1e630f · last month
3,315 Commits
.github
Fix Solaris CI after solaris-vm was upda...
4 months ago
changelog
Fix syscall_info (#2653)
last month
examples
Fix syscall_info (#2653)
last month
src
Fix syscall_info (#2653)
last month
test
Expose UDP GSO/GRO on android (#26...
last month
.cirrus.yml
ci: migrate aarch64/Linux CI to GA (#26...
8 months ago
.gitattributes
use better merge algorithm for CHANG...
8 years ago
.gitignore
30X performance improvement in with_...
3 years ago
CHANGELOG.md
chore: drop 0.30.1
5 months ago
About
Rust friendly bindings to *nix APIs
rust
unix
rust-bindings
libc
Readme
MIT license
Contributing
Activity
Custom properties
2.9k stars
28 watching
716 forks
Report repository
Releases


teletypewriter - crates.io: Rust
crates.io/crates/teletypewriter
crates.io
Type 'S' or '/' to search
Browse All Crates
Log in with GitHub
teletypewriter v2.0.1
Rust crate to create a pseudoterminal (pty) that emulates a tty, which is a command in Unix and Unix-like operating systems to print the file name of the terminal connected to standard input. tty stands for TeleTYpewriter. This project is created and maintained for Rio terminal purposes but feel free to use it.
Readme
77 Versions
Dependencies
Dependents
Teletypewriter
Rust crate to create a pseudoterminal (pty) that emulates a tty, which is a command in Unix and Unix-like operating systems to print the file name of the terminal connected to standard input. tty stands for TeleTYpewriter. This project is created and maintaned for Rio terminal purposes but feel free to use it.
Metadata
pkg:cargo/teletypewriter...
over 2 years ago
2021 edition
MIT
1 462 SLoC
18,8 KiB
Install

The State Diagram



Actions

ignore
print
execute
clear
collect
param
esc_dispatch
csi_dispatch
hook
put
unhook
osc_start
osc_put
osc_end

ground, escape, escape intermediate, csi entry, csi param, csi intermediate, csi ignore, dcs entry, dcs param, dcs intermediate, dcs passthrough, dcs ignore, osc string, sos/pm/apc string

Managing The Terminal Emulator

Terminal

1. Receives PTY updates
2. Parses it
3. Manage it’s own state

```rust
#[derive(Debug)]
pub struct Crosswords<U>
where
    U: EventListener,
{
    active_charset: CharsetIndex,
    mode: Mode,
    pub vi_mode_cursor: ViModeCursor,
    semantic_escape_chars: String,
    pub grid: Grid<Square>,
    inactive_grid: Grid<Square>,
    scroll_region: Range<Line>,
    tabs: TabStops,
    event_proxy: U,
    pub selection: Option<Selection>,
    pub colors: TermColors,
    pub title: String,
    damage: TermDamageState,
    pub graphics: Graphics,
    pub cursor_shape: CursorShape,
    pub default_cursor_shape: CursorShape,
    pub blinking_cursor: bool,
    pub window_id: WindowId,
    pub route_id: usize,
    title_stack: Vec<String>,
    pub current_directory: Option<std::path::PathBuf>,

    // The stack for the keyboard modes.
    keyboard_mode_stack: [u8; KEYBOARD_MODE_STACK_MAX_DEPTH],
    keyboard_mode_idx: usize,
    inactive_keyboard_mode_stack: [u8; KEYBOARD_MODE_STACK_MAX_DEPTH],
    inactive_keyboard_mode_idx: usize,
}
```

OSC dispatch

```
// Reset foreground color.
b"110" => self.handler.reset_color(NamedColor::Foreground as usize),

// Reset background color.
b"111" => self.handler.reset_color(NamedColor::Background as usize),

// Reset text cursor color.
b"112" => self.handler.reset_color(NamedColor::Cursor as usize),

// OSC 1337 is not necessarily only used by iTerm2 protocol
// OSC 1337 is equal to xterm OSC 50
b"1337" => {
    if let Some(graphic) = iterm2_image_protocol::parse(params) {
        self.handler.insert_graphic(graphic, None);
    }
}
```

Update Terminal State

```rust
#[inline]
fn reset_color(&mut self, index: usize) {
    // Damage terminal if the color changed and it's not the cursor.
    if index != NamedColor::Cursor as usize && self.colors[index].is_some() {
        self.mark_fully_damaged();
    }

    self.colors[index] = None;
}
```

Render

(Decide CPU or GPU path (e.g: Metal, Vulkan, DX12, OpenGL, etc.)

From Terminal User Interface Perspective

termios(3) — Linux manual page

NAME top

termios, tcgetattr, tcsetattr, tcsendbreak, tcdrain, tcflush, tcflow, cfmakeraw, cfgetospeed, cfgetispeed, cfsetispeed, cfsetospeed, cfsetspeed - get and set terminal attributes, line control, get and set baud rate

LIBRARY top

Standard C library (*libc*, *-lc*)

SYNOPSIS top

```
#include <termios.h>
#include <unistd.h>

int tcgetattr(int fd, struct termios *termios_p);
int tcsetattr(int fd, int optional_actions,
              const struct termios *termios_p);

int tcsendbreak(int fd, int duration);
int tcdrain(int fd);
int tcflush(int fd, int queue_selector);
int tcflow(int fd, int action);

void cfmakeraw(struct termios *termios_p);

speed_t cfgetispeed(const struct termios *termios_p);
speed_t cfgetospeed(const struct termios *termios_p);

int cfsetispeed(struct termios *termios_p, speed_t speed);
int cfsetospeed(struct termios *termios_p, speed_t speed);
int cfsetspeed(struct termios *termios_p, speed_t speed);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

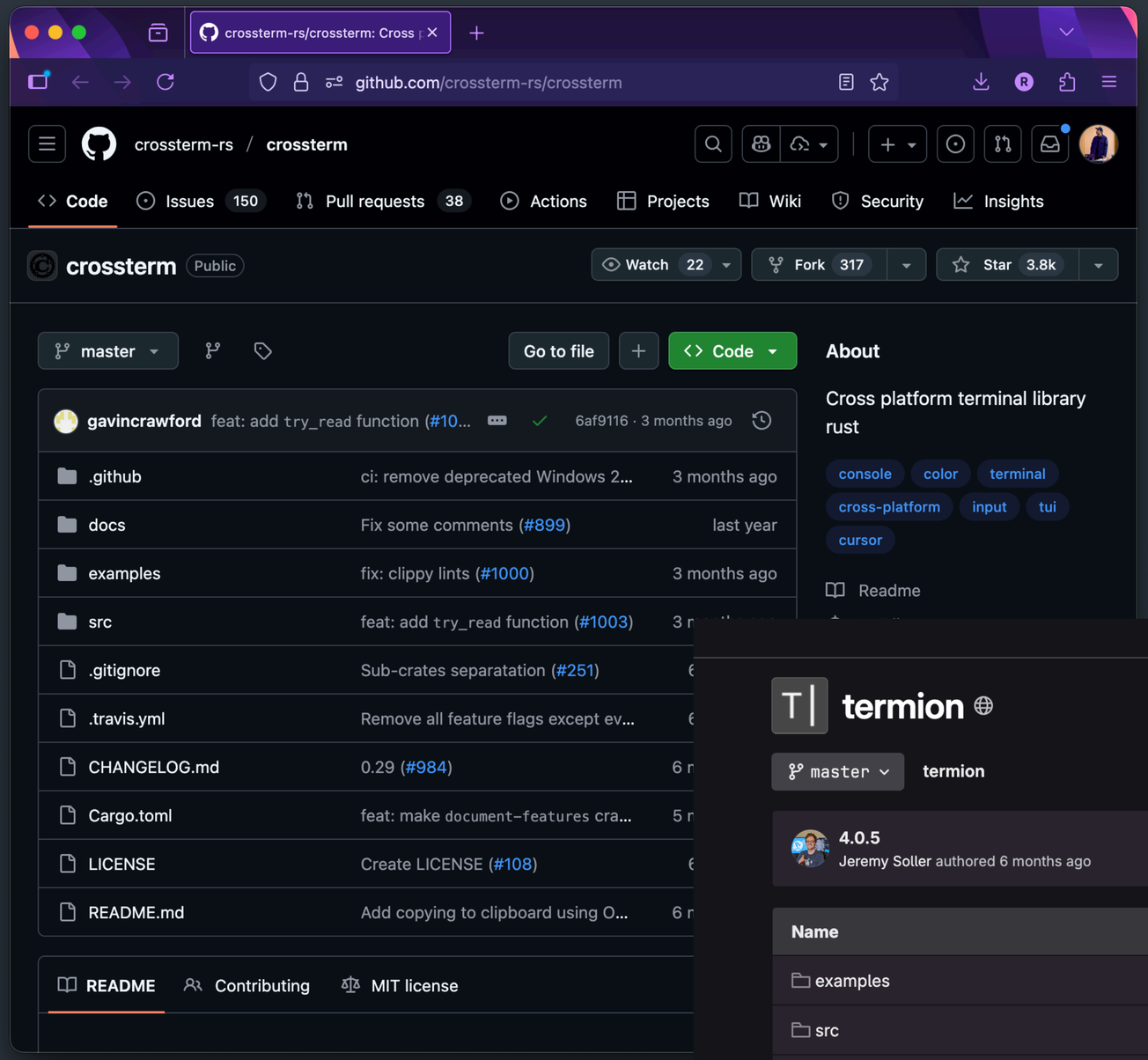
cfsetspeed(), **cfmakeraw**():
Since glibc 2.19:
_DEFAULT_SOURCE
glibc 2.19 and earlier:
_BSD_SOURCE

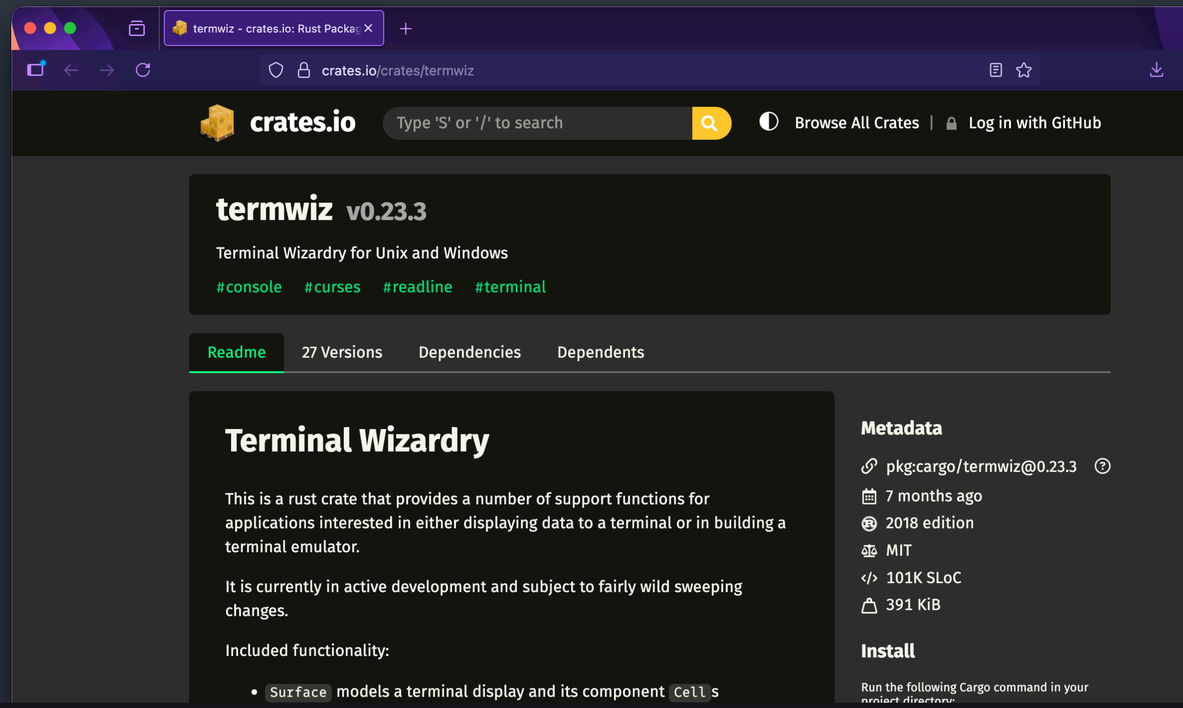
DESCRIPTION top

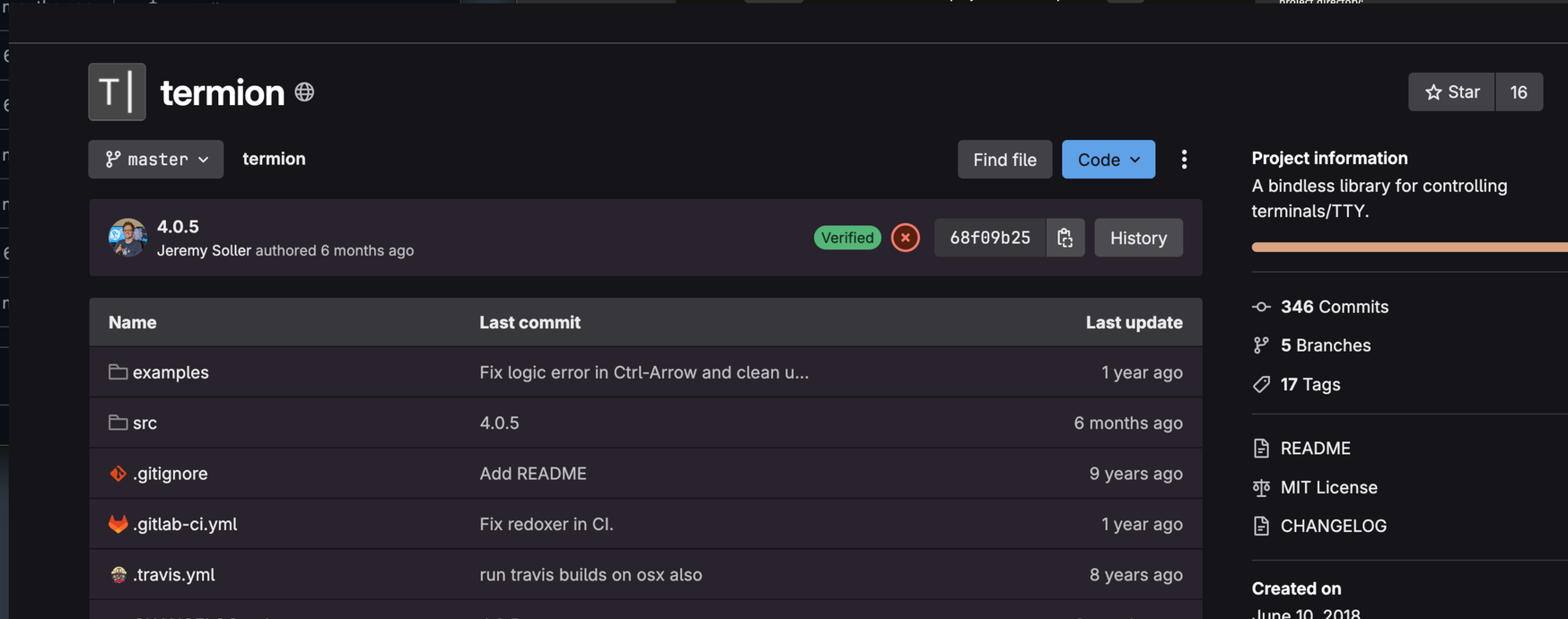
The termios functions describe a general terminal interface that is provided to control asynchronous communications ports.

The termios structure

Many of the functions described here have a *termios_p* argument that is a pointer to a *termios* structure. This structure contains at least the following members:


crossterm-rs/crossterm: Cross
github.com/crossterm-rs/crossterm
crossterm-rs / crossterm
Code
Issues 150
Pull requests 38
Actions
Projects
Wiki
Security
Insights
crossterm
Public
Watch 22
Fork 317
Star 3.8k
master
Go to file
Code
gavincrawford feat: add try_read function (#10...
6af9116 · 3 months ago
.github ci: remove deprecated Windows 2... 3 months ago
docs Fix some comments (#899) last year
examples fix: clippy lints (#1000) 3 months ago
src feat: add try_read function (#1003)
.gitignore Sub-crates separatation (#251)
.travis.yml Remove all feature flags except ev...
CHANGELOG.md 0.29 (#984)
Cargo.toml feat: make document-features cra...
LICENSE Create LICENSE (#108)
README.md Add copying to clipboard using O...
README
Contributing
MIT license
About
Cross platform terminal library rust
console
color
terminal
cross-platform
input
tui
cursor
Readme


termwiz - crates.io: Rust Packa
crates.io/crates/termwiz
crates.io
Type 'S' or '/' to search
Browse All Crates
Log in with GitHub
termwiz v0.23.3
Terminal Wizardry for Unix and Windows
#console #curses #readline #terminal
Readme
27 Versions
Dependencies
Dependents
Terminal Wizardry
This is a rust crate that provides a number of support functions for applications interested in either displaying data to a terminal or in building a terminal emulator.
It is currently in active development and subject to fairly wild sweeping changes.
Included functionality:
Surface models a terminal display and its component Cell s
Metadata
pkg:cargo/termwiz@0.23.3
7 months ago
2018 edition
MIT
101K SLoC
391 KiB
Install
Run the following Cargo command in your

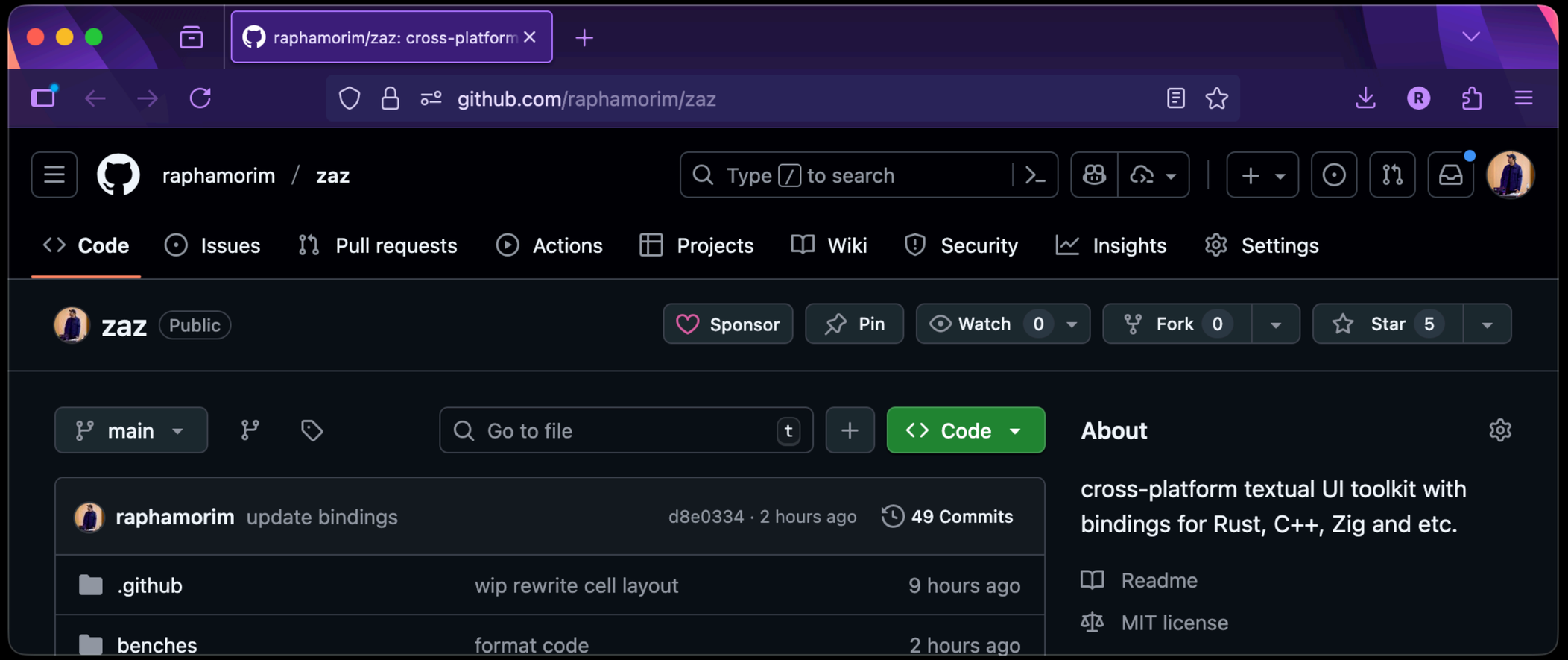

termion
Star 16
master
termion
Find file
Code
4.0.5
Jeremy Soller authored 6 months ago
Verified
68f09b25
History
Name
Last commit
Last update
examples Fix logic error in Ctrl-Arrow and clean u... 1 year ago
src 4.0.5 6 months ago
.gitignore Add README 9 years ago
.gitlab-ci.yml Fix redoxer in CI. 1 year ago
.travis.yml run travis builds on osx also 8 years ago
Project information
A bindless library for controlling terminals/TTY.
346 Commits
5 Branches
17 Tags
README
MIT License
CHANGELOG
Created on

```rust
pub(crate) struct Backend {
    original_termios: Option<Termios>,
    initialized: bool,
}

#[cfg(unix)]
use std::os::unix::io::AsRawFd;

#[cfg(unix)]
struct Termios {
    termios: libc::termios,
}

#[cfg(not(unix))]
struct Termios;

impl Backend {
    fn new() -> Self {
        Self {
            original_termios: None,
            initialized: false,
        }
    }

    pub(crate) fn init() -> Result<()> {
        let backend = BACKEND.get_or_init(|| Mutex::new(Backend::new()));
        let mut guard = backend.lock().unwrap();

        if guard.initialized {
            return Err(Error::AlreadyInitialized);
        }

        guard.enable_raw_mode()?;
        guard.initialized = true;
```

```rust
use std::io::{self, Write};
pub struct Tui;
impl Tui {
    pub fn clear() {
        print!("\x1B[2J\x1B[H");
    }
    pub fn goto(x: u16, y: u16) {
        print!("\x1B[{};{}H", y, x);
    }
    pub fn print(x: u16, y: u16, text: &str) {
        Self::goto(x, y);
        print!("{}", text);
    }
    pub fn color(fg: u8) {
        print!("\x1B[38;5;{}m", fg);
    }
    pub fn reset_color() {
        print!("\x1B[0m");
    }
    pub fn flush() {
        io::stdout().flush().unwrap();
    }
    pub fn hide_cursor() {
        print!("\x1B[?25l");
    }
    pub fn show_cursor() {
        print!("\x1B[?25h");
    }
}
```

```rust
fn main() {
    Tui::clear();
    Tui::hide_cursor();

    Tui::print(5, 2, "Hello, TUI!");

    Tui::color(196); // Red
    Tui::print(5, 4, "Colored text!");
    Tui::reset_color();

    Tui::print(5, 6, "Press any key...");
    Tui::flush();

    // Wait for input
    let mut buf = String::new();
    io::stdin().read_line(&mut buf).unwrap();

    Tui::show_cursor();
    Tui::clear();
}
```

raphamorim/zaz




cargo run --release
colors_rgb example. Press q to quit
51.3 fps

References:

- https://vt100.net/emu/dec_ansi_parser
- https://vt100.net/emu/
- github.com/charmbracelet/sequin
- github.com/charmbracelet/gum
- github.com/charmbracelet/ultraviolet
- github.com/saitoha/libsixel
- github.com/cryptocode/terminal-doom
- https://sw.kovidgoyal.net/kitty/graphics-protocol/
- https://en.wikipedia.org/wiki/Sixel
- https://github.com/raphamorim/zaz

Rio terminal stickers!

Thank you!
Obrigado!
Grazie!