

Designing TUI Applications in Rust



Orhun Parmaksız

Who am I?

- FOSS Developer
 - <https://github.com/orhun>
- Back-end Developer (Rust)
 - <https://www.linkedin.com/in/orhunp>
- Daily user of Linux, member of the Arch Linux team
 - <https://archlinux.org/people/trusted-users/#orhun>
- Software Engineering student (2/4)

<https://orhun.dev>

Roadmap

- Context
 - What is TUI?
 - Libraries (crates) for building TUIs
- Implementation
 - Building a simple TUI using tui-rs
- Showcase: gpg-tui
- Q&A

TUI?

TUI stands for text-based user interface or terminal user interface.


Text-based because primarily, you have a bunch of text on the screen and terminal user interface because they are used only in the terminal.

TUI apps are often also considered as CLI applications because they are restricted to the terminal.

<https://itsfoss.com/gui-cli-tui/>

Show me!

Search

Kernel Release 
Linux arch 5.5.8-arch1-1

Loaded Kernel Modules (1/156) (0%)

Module	Size	Used by
hid_generic	16.4 KB	0 -
snd_usb_audio	294.9 KB	0 -
snd_usbmidi_lib	41.0 KB	1 snd_usb_audio
usbhid	65.5 KB	0 -
hid	147.5 KB	2 hid_generic,usbhid
snd_rawmidi	45.1 KB	1 snd_usbmidi_lib
snd_seq_device	16.4 KB	1 snd_rawmidi
tun	57.3 KB	0 -
ipt_REJECT	16.4 KB	0 -
nf_reject_ipv4	16.4 KB	1 ipt_REJECT
xt_conntrack	16.4 KB	0 -
xt_MASQUERADE	20.5 KB	0 -
xt_CHECKSUM	16.4 KB	0 -
xt_comment	16.4 KB	0 -
xt_tcpudp	20.5 KB	0 -
iptables_mangle	16.4 KB	0 -
iptables_nat	16.4 KB	0 -

Module: hid_generic

filename:
/lib/modules/5.5.8-arch1-1/kernel/drivers/hid/hid-generic.ko.xz

license: GPL

description: HID generic driver

author: Henrik Rydberg

srcversion: AA4DBCEB2C92FCF8B805A86

alias: hid:b*g*v*p*

depends: hid

retpoline: Y

intree: Y

name: hid_generic

vermagic: 5.5.8-arch1-1 SMP preempt

mod_unload


sig_id: PKCS#7

signer: Build time autogenerated kernel key

sig_key:
65:76:9B:3C:6B:17:B4:3A:51:96:86:2E:B1:03:04:7A:FC:C9:9D:95

sig_hashalgo: sha512

signature:

Kernel Activities 

[Sat Mar 21 20:05:04 2020] wlp3s0: authenticate with 4c:9e:ff:02:b0:40

[Sat Mar 21 20:05:04 2020] wlp3s0: send auth to 4c:9e:ff:02:b0:40 (try 1/3)

[Sat Mar 21 20:05:04 2020] wlp3s0: authenticated

[Sat Mar 21 20:05:04 2020] wlp3s0: associate with 4c:9e:ff:02:b0:40 (try 1/3)

[Sat Mar 21 20:05:04 2020] wlp3s0: RX AssocResp from 4c:9e:ff:02:b0:40 (capab=0x411 status=0 aid=3)

[Sat Mar 21 20:05:04 2020] wlp3s0: associated

[Sat Mar 21 20:05:04 2020] IPv6: ADDRCONF(NETDEV_CHANGE): wlp3s0: link becomes ready

<https://github.com/orhun/kmon>

TUI examples

- <https://github.com/Rigellute/spotify-tui>
- <https://github.com/extrawurst/gitui>
- <https://github.com/kdheepak/taskwarrior-tui>
- <https://github.com/orhun/gpg-tui>
- <https://github.com/sayanarjit/xplr>

TUI backends

- termion
- rustbox
- crossterm
- pancurses
- ncurses-rs
- bearlibterminal-rs

Crates for building TUI apps

- **tui-rs**
 - <https://github.com/fdehau/tui-rs>
- **cursive**
 - <https://github.com/gyscos/cursive>
- **dialoguer**
 - <https://github.com/mitsuhiko/dialoguer>

tui-rs

- Requires rustc $\geq 1.44.0$
- Supports termion, rustbox, crossterm and pancurses backends
- It has a bunch of useful widgets:
 - Block
 - Gauge
 - Sparkline
 - Chart / BarChart
 - List
 - Table
 - Paragraph
 - Canvas
 - Tabs

Implementation

<https://github.com/orhun/rust-tui-example>

Creating the project

```
cargo new --bin rust-tui-example --vcs=git  
cd rust-tui-example/
```


Add tui-rs as dependency

```
[package]
name = "rust-tui-example"
version = "0.1.0"
edition = "2018"
```

```
[dependencies]
anyhow = "1.0"
crossterm = "0.20.0"
```

```
[dependencies.tui]
version = "0.16.0"
default-features = false
features = ["crossterm"]
```


Workflow

1. Initialize the terminal
2. Create an event handler
3. Prepare the terminal for rendering
4. Start the render loop
 1. Render widgets
 2. Handle events
5. Flush the terminal before exit

Workflow

```
1 fn main() -> Result<()> {
2     /* 1- Initialize the terminal */
3
4     /* 2- Create an event handler */
5
6     /* 3- Prepare the terminal for rendering */
7
8     /* 4- Start the render loop */
9     let mut running = true;
10    while running {
11        /* 4.1- Render widgets */
12
13        /* 4.2- Handle events */
14    }
15
16    /* 5- Flush the terminal before exit */
17
18    ok(())
19 }
```


1- initializing the terminal

```
let backend = tui::backend::CrosstermBackend::new(std::io::stderr());  
let mut terminal = tui::Terminal::new(backend)?;
```


2- creating an event handler

```
// https://github.com/fdehau/tui-rs/blob/v0.6.0/examples/util/event.rs  
// https://github.com/orhun/gpg-tui/blob/v0.7.4/src/term/event.rs  
mod event;  
  
let event_handler = event::EventHandler::new(250);
```


3- preparing the terminal for rendering

```
terminal::enable_raw_mode()?;  
crossterm::execute!(io::stderr(), EnterAlternateScreen, EnableMouseCapture)?;  
terminal.hide_cursor()?;  
terminal.clear()?;
```


4.1- rendering the widgets

```
/* 4- Start the render loop */
let mut running = true;
while running {

    /* 4.1- Render widgets */
    terminal.draw(|frame| {
        frame.render_widget(tui::widgets::Paragraph::new("rust munich"), frame.size())
    })?;

}
```


4.2- handling the key press events

```
/* 4- Start the render loop */
let mut running = true;
while running {

    /* 4.1- Render widgets */
    terminal.draw(|frame| {
        frame.render_widget(tui::widgets::Paragraph::new("rust munich"), frame.size())
    })?;

    /* 4.2- Handle events */
    match event_handler.next()? {
        Event::Key(key_event) => match key_event.code {
            // exit on ESC key press
            KeyCode::Esc => {
                running = false;
            }
            _ => {}
        },
        _ => {}
    }
}
```


5- flushing the terminal before exit

```
terminal::disable_raw_mode()?;  
crossterm::execute!(io::stderr(), LeaveAlternateScreen, DisableMouseCapture)?;  
terminal.show_cursor()?;
```


Running our TUI program

```
cargo run --release
```

```
rust munich
```


GPG-TUI

Manage your GnuPG keys with ease!

<https://github.com/orhun/gpg-tui>

gpg-tui aims to ease the key management operations such as:

- list
- export
- sign

It provides a terminal interface along with the command-line fallback for more complex operations and tries to bring a more interactive approach to key management.

Listing Keys/Subkeys/Signatures

```
> [sc-a] rsa4096/54C28F4FF5A1A949 [f] David Runge <dave@sleepmap.de>
|   └─(2012) -> (2022)
[--e-] rsa4096/43E14A34A7571AB2
|   └─(2012) -> (2022)
|   [10] 3348882F6AC6A4C2 Pierre Schmitz (Arch Linux Master Key) <pi..
|   [10] BA1DFB64FFF979E7 Allan McRae (Arch Linux Master Key) <allan..
|   [10] 9B729B06A680C281 Bartłomiej Piotrowski (Arch Linux Master K..
|   [10] A88E23E377514E00 Florian Pritz (Arch Linux Master Key) <flo..
|   [13] selfsig (2017-05-07)
- [?] Deviser <deviser@frqrec.com>
|   └─[30] selfsig (2015-02-03) [rev]
- [f] David Runge <dave@c-base.org>
|   [10] BA1DFB64FFF979E7 Allan McRae (Arch Linux Master Key) <al..
|   [10] 9B729B06A680C281 Bartłomiej Piotrowski (Arch Linux Maste..
|   [10] A88E23E377514E00 Florian Pritz (Arch Linux Master Key) <..
|   [13] selfsig (2017-05-07)
- [?] Das Bluul <dasbluul@frqrec.com>
|   └─[30] selfsig (2015-02-03) [rev]
- [?] Wasserturm <wasserturm@frqrec.com>
|   └─[30] selfsig (2015-02-03) [rev]
- [?] David Runge <david.runge@frqrec.com>
|   └─[30] selfsig (2015-02-03) [rev]
- [f] David Runge <runge@pool.math.tu-berlin.de>
|   [10] BA1DFB64FFF979E7 Allan McRae (Arch Linux Master Key) <al..
|   [10] 9B729B06A680C281 Bartłomiej Piotrowski (Arch Linux Maste..
|   [10] A88E23E377514E00 Florian Pritz (Arch Linux Master Key) <..
|   [13] selfsig (2017-05-07)
- [?] Drafted To Haunt <draftedtohaunt@frqrec.com>
|   └─[30] selfsig (2015-02-03) [rev]
...

```

Options Menu for Key Management

```
> [sc-a] rsa4096/54C28F4FF5A1A949 [f] David Runge <dave@sleepmap.de>
|   └─(2012) → (2022)
|--e-] rsa4096/43E14A34A7571AB2
|   └─(2012) → (2022)
|   └─[10] 334882F6AC6A4C2 Pierre Schmitz (Arch Linux Master Key) <pi..
|   └─[10] BA1DFB64FFF979E7 Allan McRae (Arch Linux Master Key) <allan..
|   └─[10] 9B729B06A680C281 Bartłomiej Piotrowski (Arch Linux Master K..
|   └─[10] A88E23E377514E00 Florian Pritz (Arch Linux Master Key) <flo..

Options
refresh application
refresh the keyring
import key(s) from a file
import key(s) from clipboard
receive key(s) from keyserver
export the selected key (pub)
export all the keys (pub)
delete the selected key (pub)
send key to the keyserver
edit the selected key
sign the selected key
> generate a new key pair
enable armored output
copy exported key
copy key id
copy key fingerprint

v]
cRae (Arch Linux Master Key) <al..
iej Piotrowski (Arch Linux Maste..
Pritz (Arch Linux Master Key) <..

v]
om>
v]
.com>
v]
-berlin.de>
cRae (Arch Linux Master Key) <al..
iej Piotrowski (Arch Linux Maste..
Pritz (Arch Linux Master Key) <..

└─[13] selfsig (2017-05-07)
└─[?] Drafted To Haunt <draftedtohaunt@frqrec.com>
└─[30] selfsig (2015-02-03) [rev]
...
```

< list pub (33/126) >

GPG fallback

```
(orhun λ ~) gpg-tui --homedir /etc/pacman.d/gnupg --style colored
gpg: WARNING: unsafe ownership on homedir '/etc/pacman.d/gnupg'
Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 3072
Requested keysize is 3072 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0) ☐
```

Details – Help

[o] [space] [enter]
└show options

[h j k l] [arrows] [pgkeys]
└navigate

[n]
└switch to normal mode

[v]
└switch to visual mode

> [c]
└switch to copy mode

x: Copy the exported key
i: Copy the key id
f: Copy the key fingerprint
u: Copy the user id
1,2: Copy the content of the row
:copy

```
.ydh/ +mdh: :hdy.  
sm` ym +m- sd`hy ys  
sm` --- /h-`sy`yy` :--  
oh`/sh /hss- hy./mh  
+h-`sy /h. hd.-my  
:oo+. -o. `+sy+`-tui
```

Manage your GnuPG keys with ease! (0.7.4)
Author: Orhun Parmaksiz <orhunparmaksiz@gmail.com>
Homepage: <https://github.com/orhun/gpg-tui>

GPGME version: 1.16.0
GPGME protocol: OpenPGP
GPGME engine: "/usr/bin/gpg"
GPGME engine version: 2.2.29 (>1.4.0)
GnuPG home directory: "/etc/pacman.d/gnupg"
GnuPG data directory: "/usr/share/gnupg"
Output directory: "/etc/pacman.d/gnupg/out"
Default signing key: not specified
Armored output: false

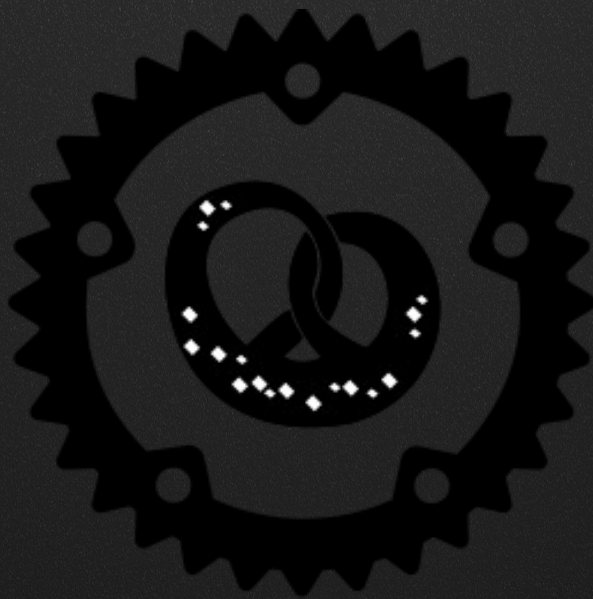
Future goals

- Sign/encrypt files (#28)
- Integrate with CLI mail clients (#19)
- Add a menu for editing keys (#16)
- Remappable key bindings (#6)
- Test it with physical devices (e.g. YubiKey, Nitrokey)
- Customizable themes

Feel free to contribute and submit your ideas!

<https://github.com/orhun/gpg-tui/issues>

Thank you!



<https://orhun.dev>