X11 and Wayland

A tale of two implementations

X11 and Wayland

A tale of two implementations

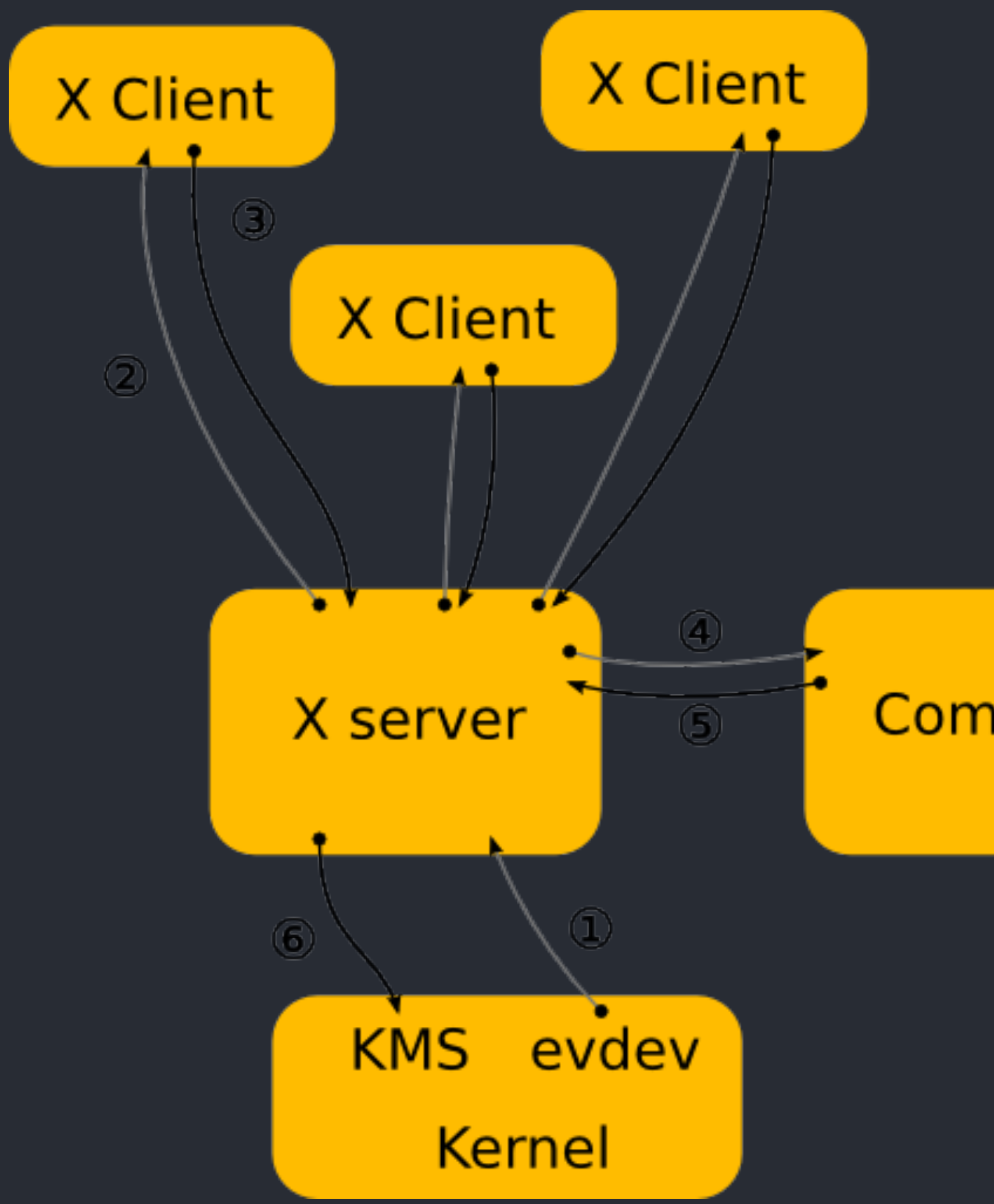
Concepts and Goals

What is hikari and what am I trying to achieve?

- window manager / compositor
- started 1.5 years ago
- written from scratch
- stacking / tiling hybrid approach inspired by
- tiling algorithm inspired by herbstluftwm
- keyboard driven, for fast navigation
- modal, inspired by vim
- waste little screen space
- allows to arbitrarily group windows
- minimal dependencies
- energy efficient
- target FreeBSD 😈
- **X11** and **Wayland** implementation


VS

X Window System Architecture



```c
// TinyWM is written by Nick Welch <nick@incise.org> in 2005 & 2011.
//
// This software is in the public domain
// and is provided AS IS, with NO WARRANTY.
#include <X11/Xlib.h>

#define MAX(a, b) ((a) > (b) ? (a) : (b))

int main(void)
{
    Display * dpy;
    XWindowAttributes attr;
    XButtonEvent start;
    XEvent ev;

    if(!(dpy = XOpenDisplay(0x0))) return 1;

    XGrabKey(dpy, XKeysymToKeycode(dpy, XStringToKeysym("F1")), Mod1Mask,
            DefaultRootWindow(dpy), True, GrabModeAsync, GrabModeAsync);
    XGrabButton(dpy, 1, Mod1Mask, DefaultRootWindow(dpy), True,
            ButtonPressMask|ButtonReleaseMask|PointerMotionMask, GrabModeAsync, GrabModeA
    XGrabButton(dpy, 3, Mod1Mask, DefaultRootWindow(dpy), True,
            ButtonPressMask|ButtonReleaseMask|PointerMotionMask, GrabModeAsync, GrabModeA

    start.subwindow = None;
    for(;;) {
        XNextEvent(dpy, &ev);
        if(ev.type == KeyPress && ev.xkey.subwindow != None)
            XRaiseWindow(dpy, ev.xkey.subwindow);
        else if(ev.type == ButtonPress && ev.xbutton.subwindow != None) {
            XGetWindowAttributes(dpy, ev.xbutton.subwindow, &attr);
            start = ev.xbutton;
        } else if(ev.type == MotionNotify && start.subwindow != None) {
            int xdiff = ev.xbutton.x_root - start.x_root;
            int ydiff = ev.xbutton.y_root - start.y_root;
            XMoveResizeWindow(dpy, start.subwindow,
                attr.x + (start.button==1 ? xdiff : 0),
                attr.y + (start.button==1 ? ydiff : 0),
                MAX(1, attr.width + (start.button==3 ? xdiff : 0)),
                MAX(1, attr.height + (start.button==3 ? ydiff : 0)));
        } else if(ev.type == ButtonRelease)
            start.subwindow = None;
    }
}
```

Talking to the X Server

Xlib

W-----RW-----RW-----RW-----R

XCB

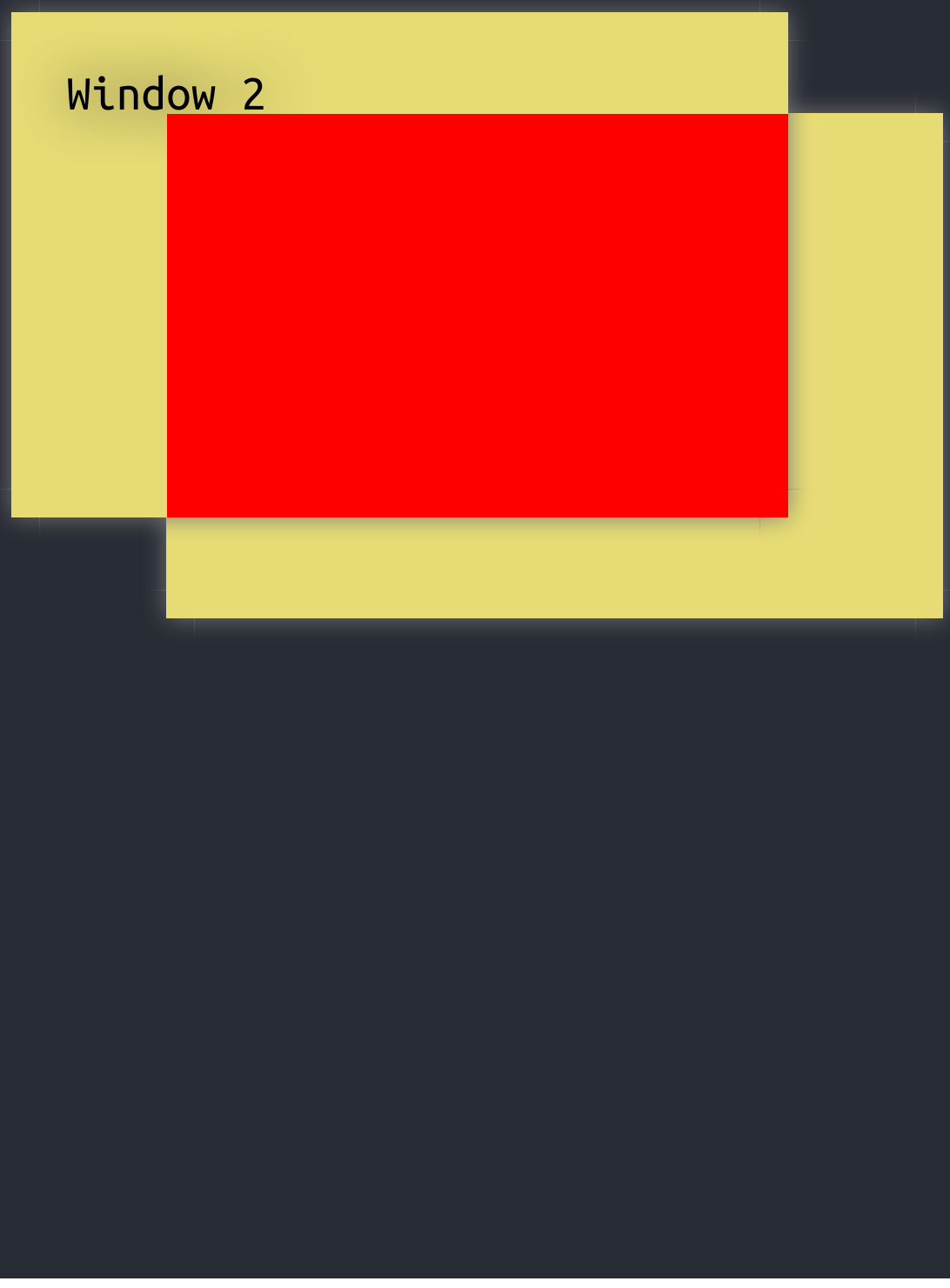
WWWW--RRRR

- W: Writing request
- -: Stalled, waiting for data
- R: Reading reply

Window ordering

Screen Artifacts

Screen Artifacts



I can haz keyboardz plz?

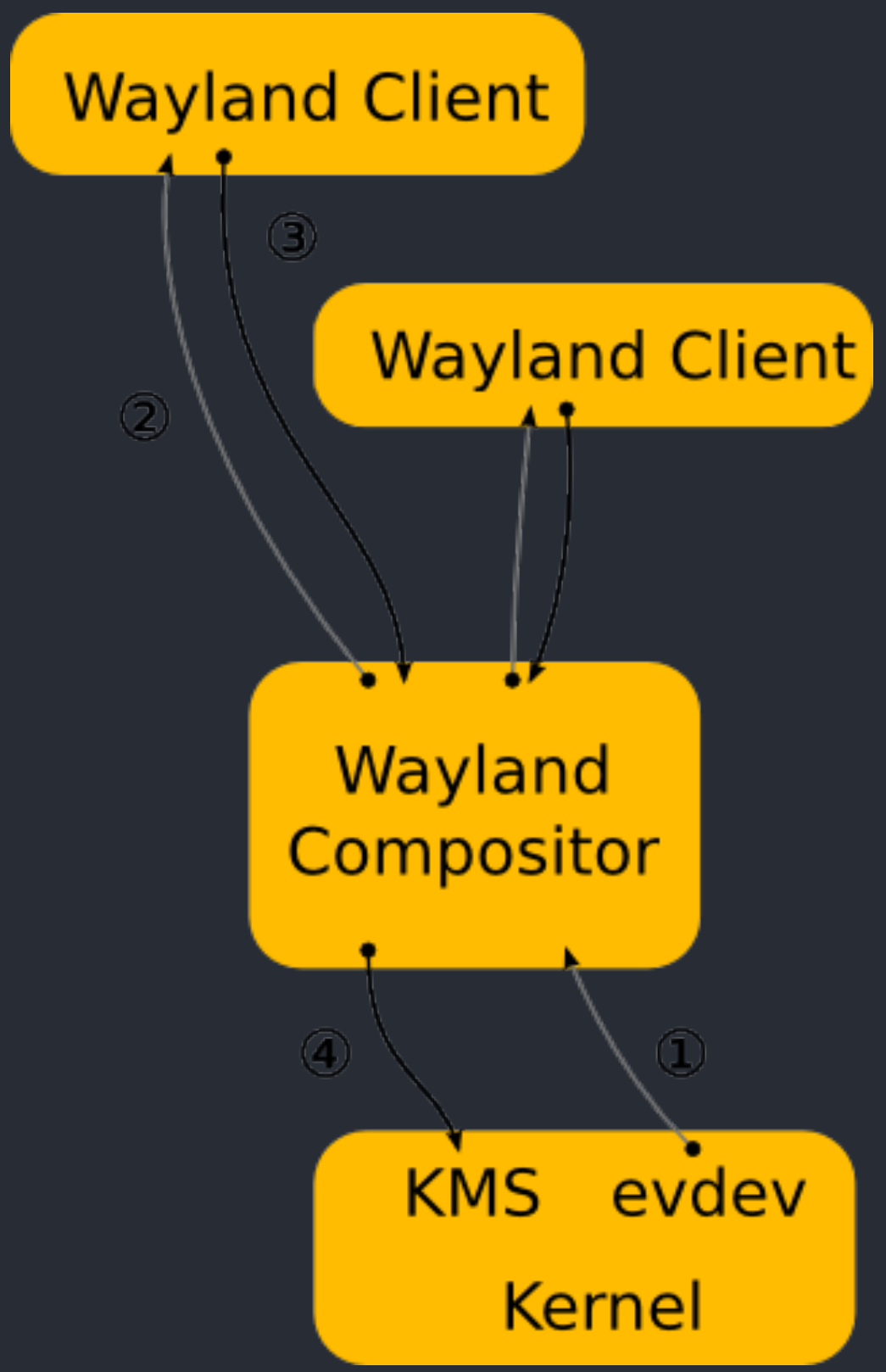
```c
// taken from awesome keygrabber.c
static bool
keygrabber_grab(void)
{
    xcb_grab_keyboard_reply_t *xgb;

    for(int i = 1000; i; i--) {
        if((xgb = xcb_grab_keyboard_reply(globalconf.conn
          xcb_grab_keyboard(globalconf.connection, true,
            globalconf.screen->root,
            XCB_CURRENT_TIME, XCB_GRAB_MODE_ASYNC,
            XCB_GRAB_MODE_ASYNC),
          NULL))) {
            p_delete(&xgb);
            return true;
        }
        usleep(1000);
    }
    return false;
}
```

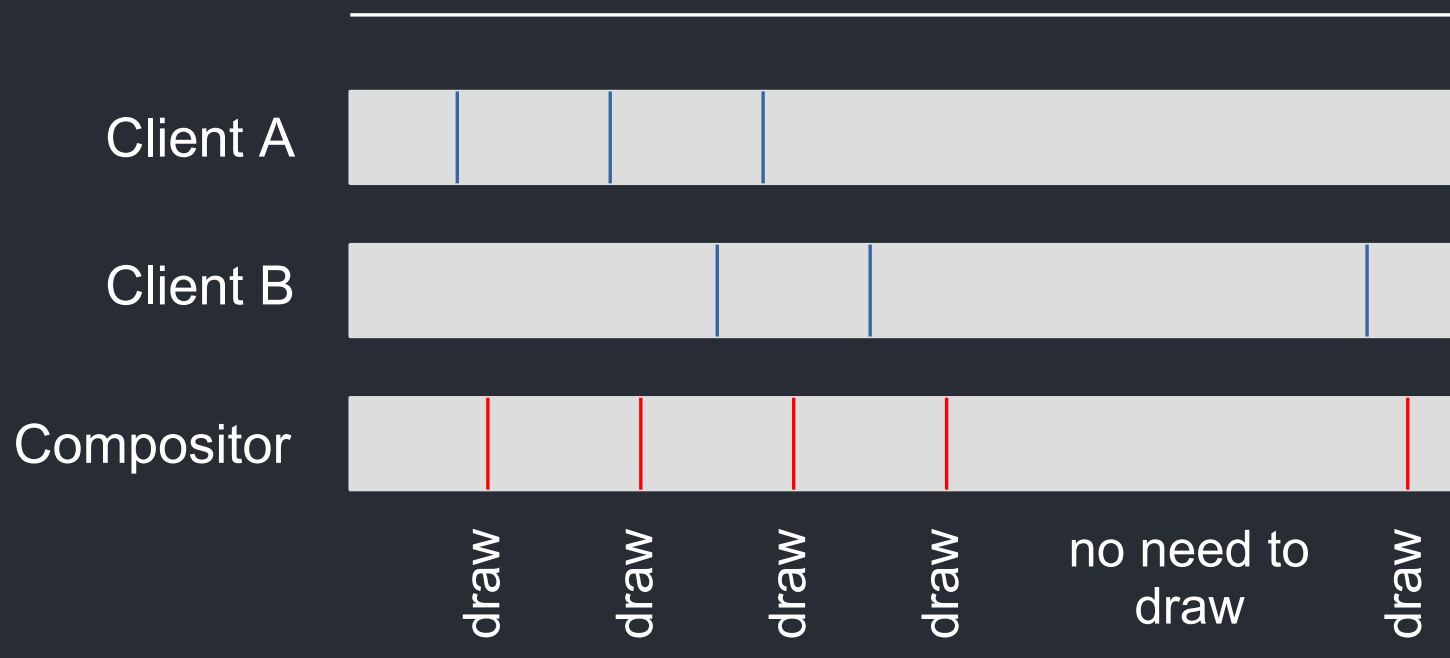
Conclusion

- very easy to get something up and running
- graphical user interfaces have evolved
- "gazillions" of X extensions (legacy demands
- global name space (bad security implication
- window manager is just a client
- duplicating functionality in the window mar
- screen artifacts (gets a bit better with COMP

Wayland Architecture



Every frame is perfect!



[1] https://emersion.fr/blog/2019/intro-to-damage

wlroots

> Pluggable, composable, unopinionated modules for Wayland compositor; or about 50,000 lines of code you to write anyway.

[2] https://github.com/swaywm/wlroots

- written in C
- used by `sway` [3] https://swaywm.org/
- `0.1` release Oct 21, 2018
- provides a common ground for many compo

Interesting compositors based on wlroots

- `tinywl` ~1KLOC (shipped with `wlroots`)
- `cage` [4] https://www.hjdskes.nl/projects/cage

Toolkits

- GTK `GDK_BACKEND=wayland`
- Qt `QT_QPA_PLATFORM=wayland-egl`
- Clutter `CLUTTER_BACKEND=wayland`
- SDL `SDL_VIDEODRIVER=wayland`

Applications

- Firefox / Thunderbird `MOZ_ENABLE_WAYLA`
- `mpv`
- `wl-clipboard` (makes my `neovim` happy)

Running X Applications on Waylan

- `Xwayland` (needs compositor support)

Conclusion

- it's harder to get something up and running
- slightly more code to have the same function had with X11
- fewer processes involved (no duplicated fun
- UI isolation
- way less complexity
- direct control over devices
- control over frames (no flickering, no tearin flashes)
- client side decorations
- more responsibility on the compositor
- large toolkit support
- great opportunity for Open Source systems t

THE C PROGRAMMING LANGUAGE

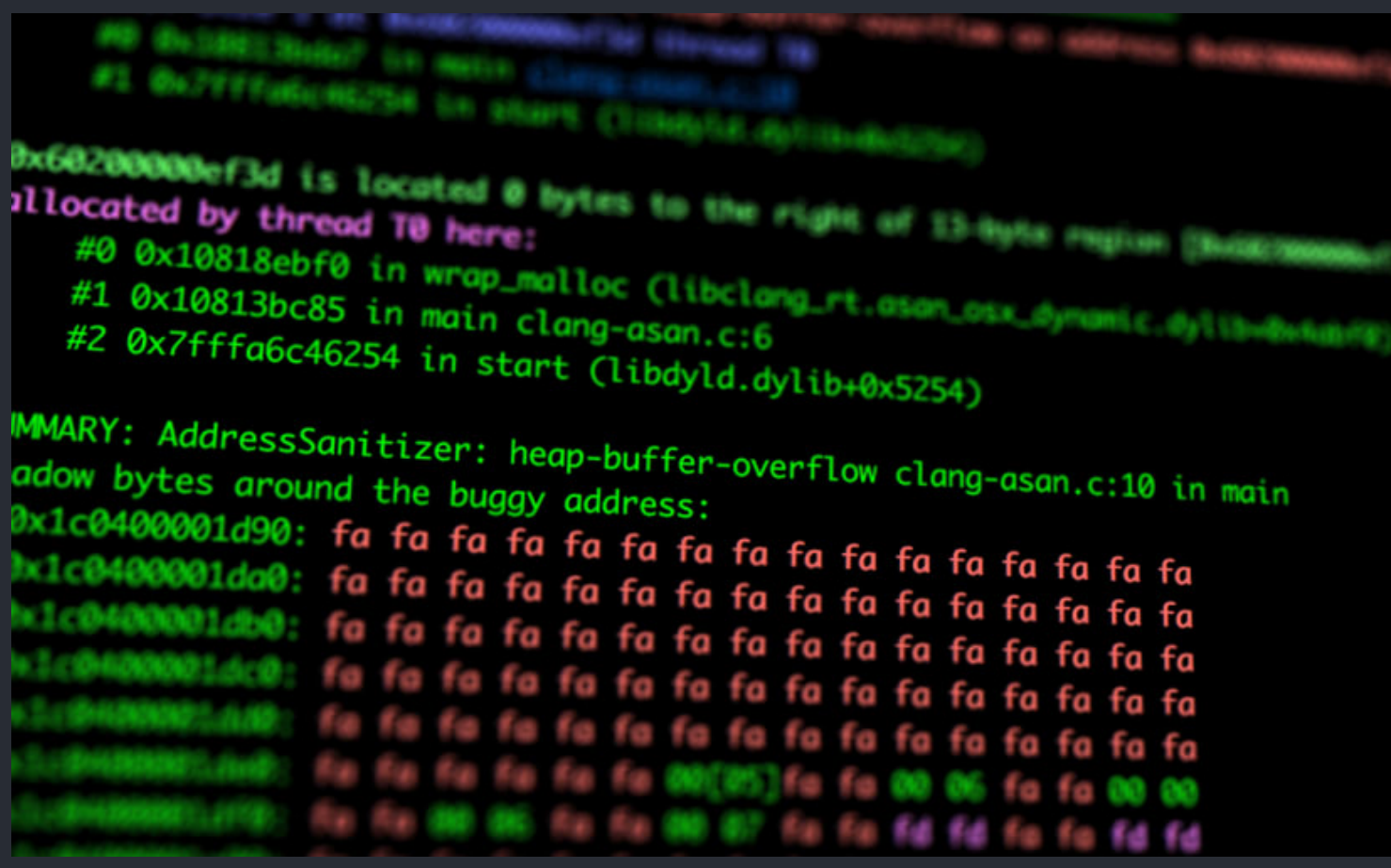
Y U NO RUST?

> The compositor part of Way Cooler is now written in C.
> portion (i.e. the side that implements the Aw
> functionality) is still written in Rust.
>
> Ultimately, wlroots-rs was too difficult to write. T
> overhead of attempting to wrap complicated C libraries w
> too demanding. This complexity often leads to a Riil
> which I am strongly against. So, the compositor is now w

[5] https://github.com/way-cooler/way-cooler/pul

ASAN

clang -fsanitize=address



Thank you!

Contact

- Mastodon: chaos.social/@raichoo
- Matrix: @raichoo:acmelabs.space
- Hikari Matrix Chat: #hikari:acmelabs.space