Adventures in porting a Wayland Compositor to NetBSD and OpenBSD

Jeff Frasca

BSDCan 2025

What is Wayland?

- Maintained by freedesktop.org
- Rethinking of GUI display protocol for modern graphics hardware
- Focus on performance for local applications
- Removed legacy features

What is Wayland?

Uses the modular Xorg ecosystem as a parts bin

- MesaLib and DRM
- xkbcommon

Unfortunately, also a lot of Linuxisms

Wayland Security

- Wayland uses Unix domain sockets by default
- graphical data gets moved around with DMA-BUFs
- Protocol has a better story about keyloggers
- Accessibility problems
- Screen locks are very non-uniform, and I suspect jwz is going to end up laughing at us

Wayland Terminology

- Compositor, Display Server
- surface, view (synonyms for "window")
- seat (keyboard-mouse-screen abstraction)
- Wayland (protocol, library, ecosystem)

Will you HAVE to use it?

- This is a talk at a BSD Conference, the answer is:

  NO.

- We vendor X11 into our OSes, so there is some amount of maintenance that doesn't rely on freedesktop.org
- Wayland support means better graphics acceleration for X11 users!
- X11 needed for GUIs on supported retro hardware

Working with the Wayland Protocol

- Base protocol is very simple and designed to be extended
- It's really a protocol development framework
- Extensions are specified in XML
- Core protocols are actually extensions
- `wl_compositor`, `wl_display` and others are extensions that are shipped with the core wayland library.

Working with the Wayland Protocol

Example protocol definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<protocol name="xuake_control_v1">
    <interface name="xuake_control" version="1">
        <description summary="xuake control proto">
        This is a protocol used by xuake to allow a client to send commands
        programmatically.
        </description>
        <request name="about">
            <description summary="Get basic info about compositor">
            Get short version info and blurb text from the compositor
            Essentially a heavy-weight ping.
            </description>
        </request>
        <event name="about_info">
            <description summary="Response event for about request">
            Sends short blurb with version info to the client.
            </description>
            <arg name="info" type="string" summary="about blurb" />
        </event>
        <request name="exit">
            <description summary="Cause xuake to exit">
            Request compositor terminates
            </description>
        </request>
    </interface>
</protocol>
```

My Compositor: Xuake

This space intentionally
left blank

(I don't have a logo)

The Good:

- Stacking Window Manager
- Minimalist, Keyboard+CLI forward, built-in terminal emulator
- Command line control tool: `xk`
- Embedded Lua interpreter used for configuration and event driven scripting
- AwesomeWM/dwm desktop tags

My Compositor: Xuake

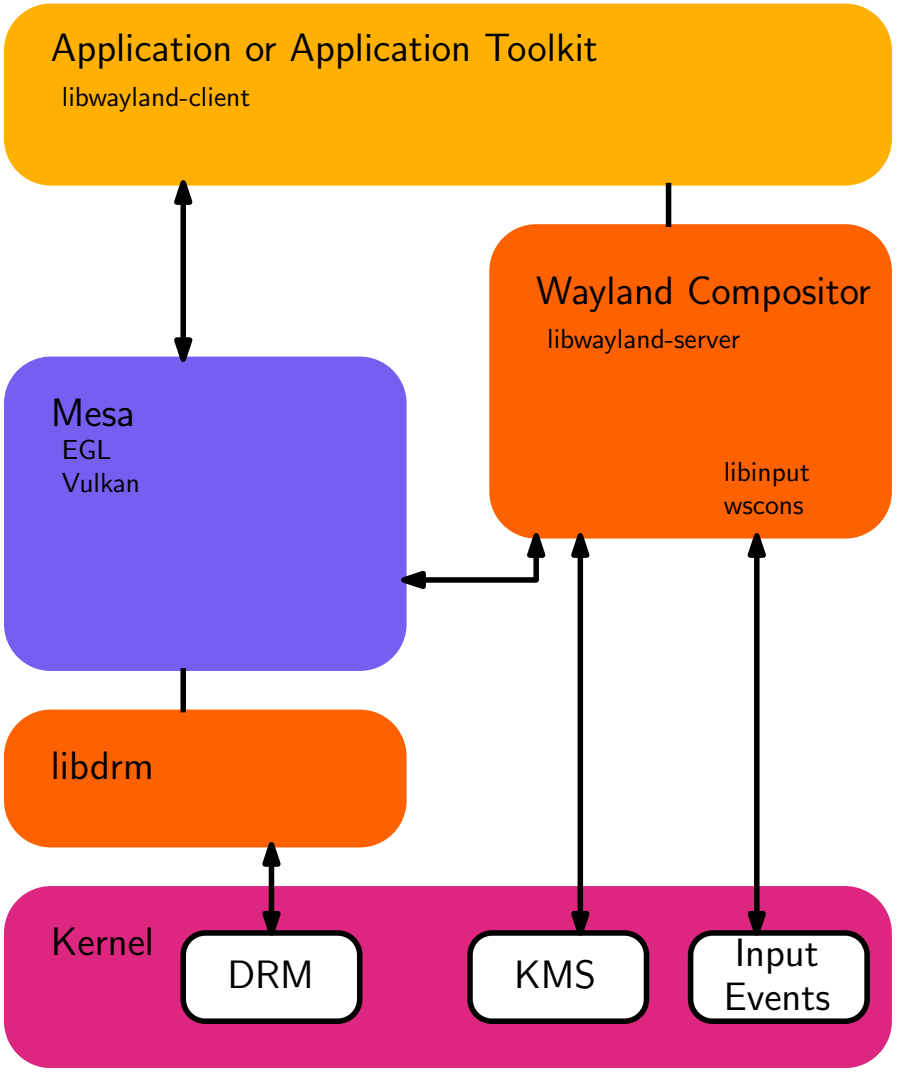
This space intentionally left blank

(I don't have a logo)

The Bad:

- Perpetual Alpha Software
- No lock screen/screen-saver
- Output hotplug is hella broken
- No drag-n-drop support (the developer is a weirdo)
- Lua API+callbacks are wildly incomplete
- Don't turn on all the compiler warnings
- ...and so much more

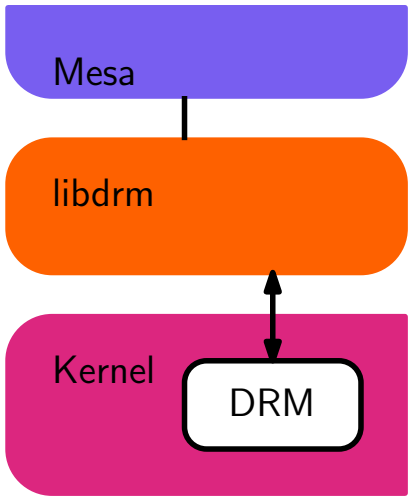
Graphics Subsystem Diagram



In Defense of `ioctl(2)`

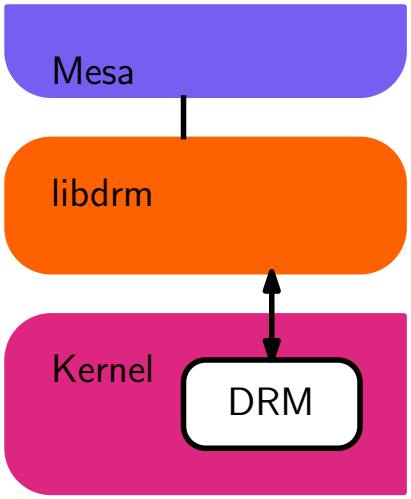
- The Filesystem is the UI abstraction of a Unix Kernel
- Most hardware has natural mappings for normal file operations
- Some hardware needs a driver-specific API
- `ioctl(2)` is how we do custom driver APIs
- It keeps the generic syscall interface very simple
- Just think of the request macro as the name of the API call

DRM API



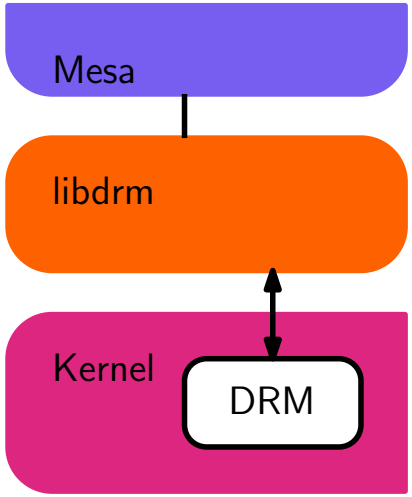
- Modern graphics hardware does not have simple mappings to file operations
- Direct Rendering Manager API provides a rich set of `ioctl(2)`s for
  - hardware configuration
  - video memory management
  - drawing commands
  - and other stuff
- API methods are per-card and granular

DRM_MASTER vs Render Nodes



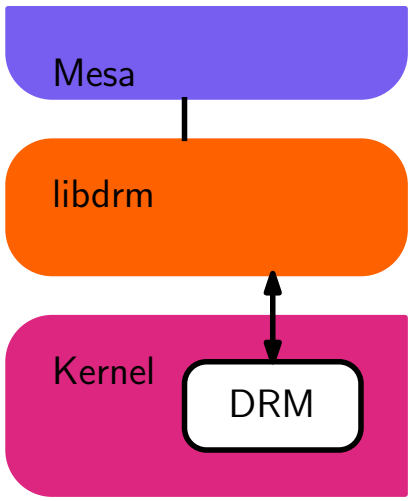
- `/dev/dri/card0`
  OpenBSD major: 87; minor: 0
- `/dev/dri/renderD128`
  OpenBSD major: 87; minor: 128
- Privilege separation mechanism
- only one process is `DRM_MASTER`
- Render Nodes are for unprivileged processes for drawing

DRM API on BSD



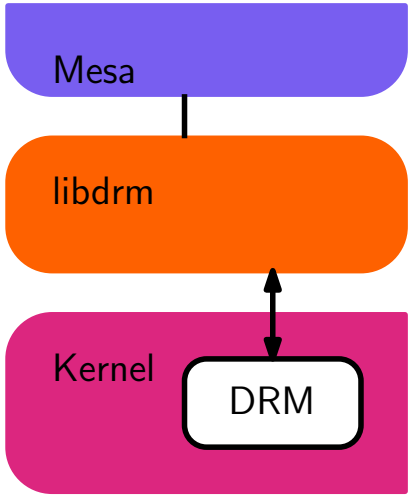
- DRM source is pulled from the Linux kernel
- Each BSD kernel has a bunch of shim code added
- Each kernel does it a little differently
- Not all data structures are fully used
- Extra `ioctl(2)` calls needed for libdrm

libdrm



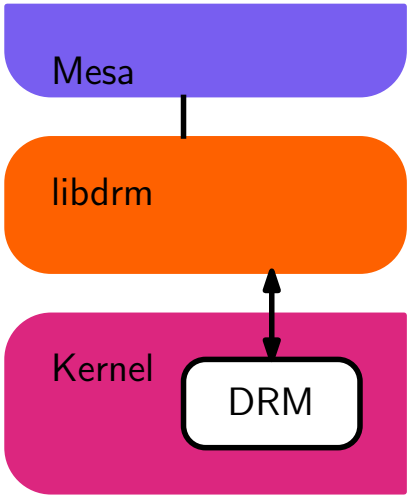
- Userspace front-end to the DRM API
- system-wide rather than per-card
- organized as larger, logical operations
- libdrm functions often make multiple `ioctl(2)` calls
- Has driver specific sub-libraries, eg, `libdrm-radeon`

libdrm on BSD



- `#ifdef`, `#ifdef` everywhere!
- libdrm on Linux heavily leverages procfs and sysfs
- Most of the extra DRM `ioctl(2)` calls provide information Linux exposes via sysfs
- Because every BSD does things a little different for the DRM shims in the kernel...
- ...the libdrm calls often have unique implementations for each BSD

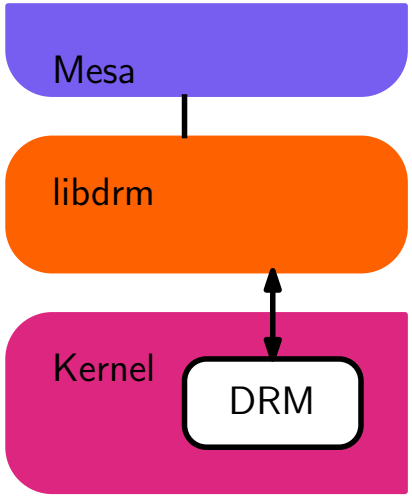
libdrm on BSD – drmParsePciBusInfo



An Example: `drmParsePciBusInfo`

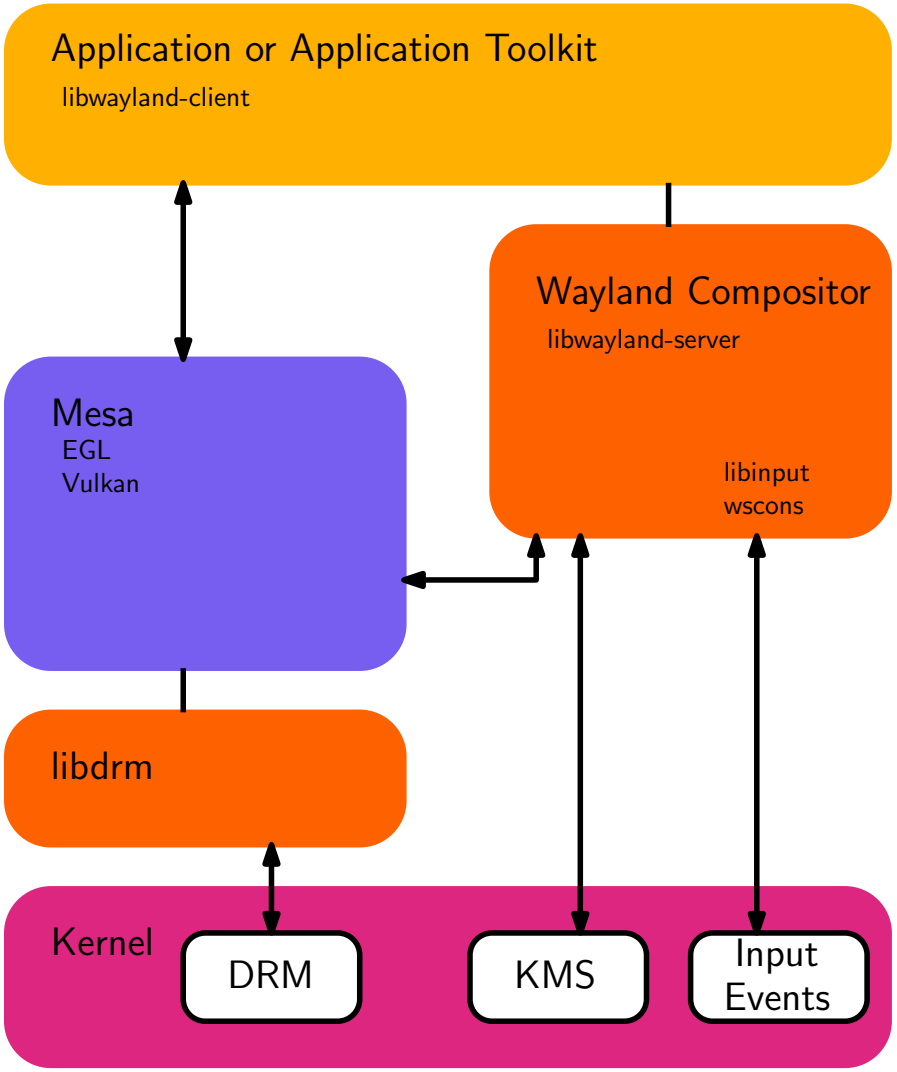
- Gets PCI domain, bus, device and function for the GPU
- used by Mesa during initialization
- Linux uses sysfs
- FreeBSD uses a sysctl
- OpenBSD and DragonflyBSD add `DRM_IOCTL_GET_PCIINFO`
- NetBSD extends `DRM_IOCTL_GET_UNIQUE` to work on render nodes

libdrm on BSD

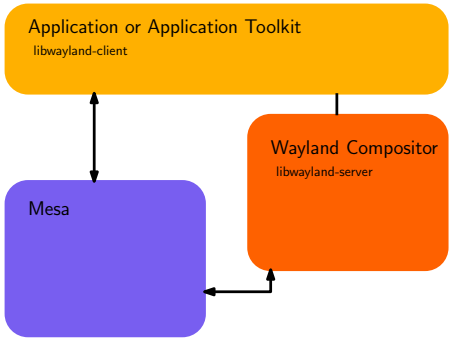


Should things be harmonized?

Graphics Subsystem Diagram

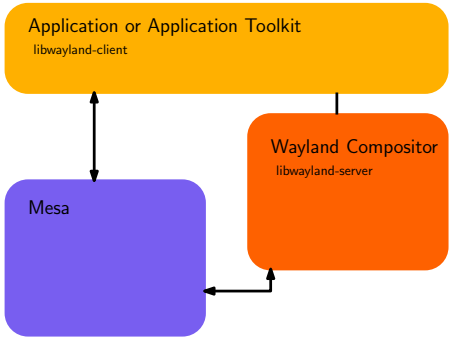


Mesa and Wayland-EGL



- The compositor initializes EGL to use `platform_drm`
- Applications that use EGL for rendering need `platform_wayland`
- `platform_wayland` causes Mesa to depend on libwayland
- OpenBSD and NetBSD don't include `platform_wayland` in the vendored Mesa

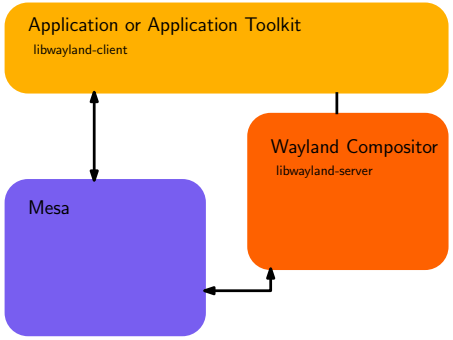
Mesa and Wayland-EGL: NetBSD



On NetBSD, there's a straightforward path to get `platform_wayland` support in libEGL

- Do not install the X11 sets
- Add `X11_TYPE=modular` to mk.conf
- Build `graphics/MesaLib` using pkgsrc

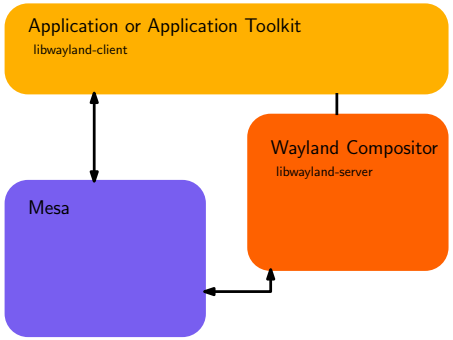
Mesa and Wayland-EGL: NetBSD



But there are some catches

- Caveat 1: MesaLib in pkgsrc is on the Amber branch for compatibility with old hardware
- Caveat 2: This only works with -current; the extension to `DRM_IOCTL_GET_UNIQUE` didn't make it into NetBSD 10
- Caveat 3

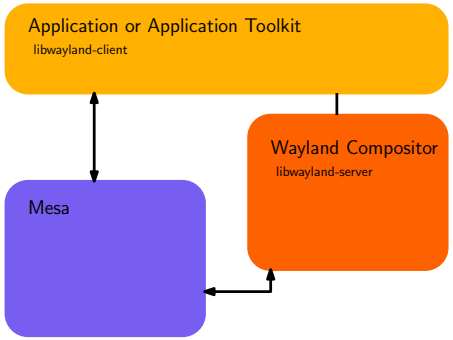
Mesa and Wayland-EGL: OpenBSD



OpenBSD 7.7 is both more and less straightforward, first, the more side:

- There are binary packages for libwayland (and sway and wlroots)
- The toolkit libraries in ports are compiled with wayland support!
- Xuake compiled against a fresh OpenBSD install with dependencies from ports on 7.7
- Qt apps from ports worked once the Qt wayland plugin was installed

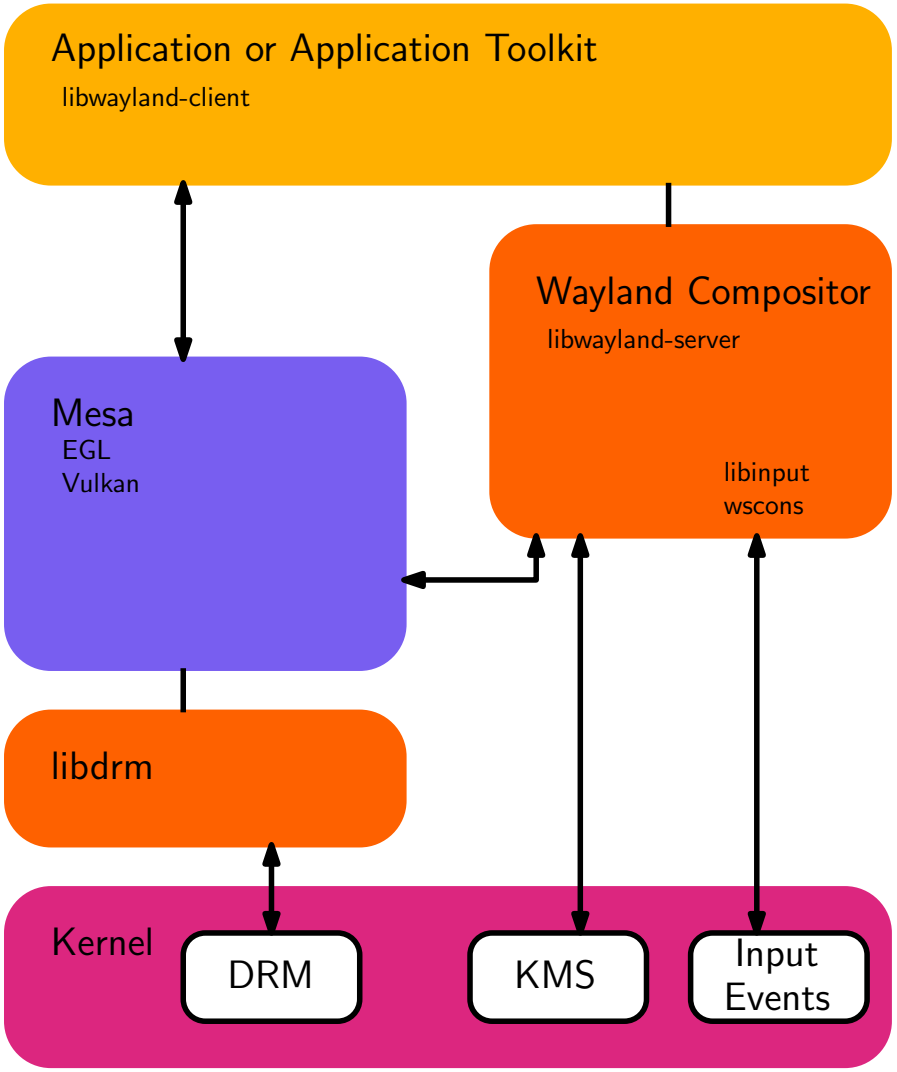
Mesa and Wayland-EGL: OpenBSD



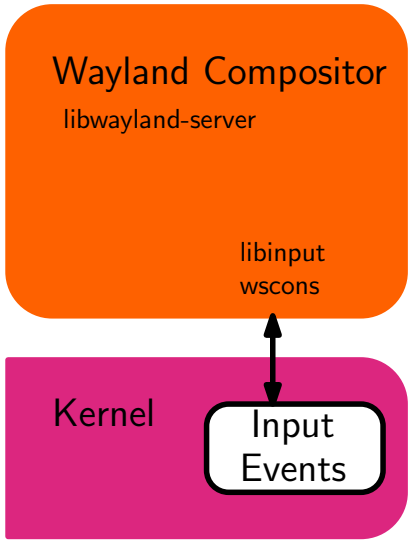
And the less straightforward side:

- `xkterm`, my Wayland-EGL based terminal emulator, crashed on startup
- Xenocara uses a lightly modified Mesa 23.3.6
- it wasn't too hard to patch the stock Mesa tarball and install over the files from xbase
- This can also be solved with pkgsrc, using exactly the steps for NetBSD

Graphics Subsystem Diagram

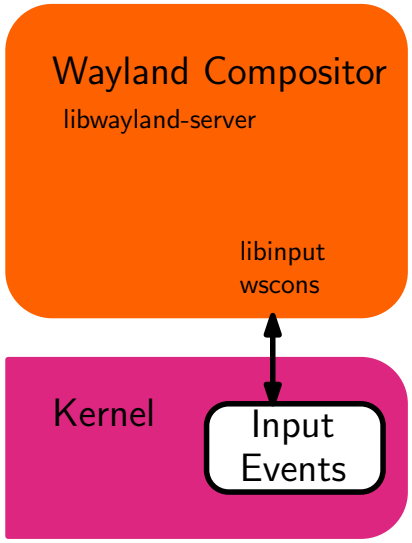


Input Events – libinput



- Linux rewrite of its input handling system
- designed to handle hotplugging a broad array of input devices
- has ports to FreeBSD, OpenBSD and DragonflyBSD
- not ported to NetBSD
- Overengineered, IMO
- required by all compositors besides Xuake and swc

Input Events – wscons



- NetBSD and OpenBSD's console system
- includes input event multiplexers
- Simple. Possibly underengineered
- Xuake's wscons backend follows Nia's swc port
- Mixed PC and USB keyboard bug

wlroots on NetBSD

- Needs a ‘wscons‘ backend
- Needs a simplified seat backend
- Examples of both of these are in the vendored version of wlroots in Xuake
- Could use similar approach as pkgsrc `kqueue(2)` patch for `devel/wayland`

Thank You

- Taylor R. Campbell
- Nia Alarie
- OpenBSD dev team
- OpenBSD ports maintainers, especially the wayland packages
- NetBSD dev team
- pkgsrc maintainers
- freedesktop.org
- sway and wlroots teams
- BSDCan Organizers and Volunteers

Future work

- OpenBSD port that doesn't need separately built Mesa
- NetBSD AMDGPU debugging
- pkgsrc build for Xuake
- wscons mixed keyboard fixes
- Optional modern MesaLib package for pkgsrc
- Lots of feature work in Xuake itself
- wscons cleanup bugs in OpenBSD
- DRM shutdown code missing in NetBSD
- file bugs instead of surprising devs in a conference talk

Questions

Thank you for coming to my talk!

Any questions?

Slides created with LaTeX on OpenBSD+Xuake
Run BSD logo pirated from `runbsd.info`

Mastodon: @overeducatedredneck@bitbang.social