# Wayland Book

## Introduction

Wayland is the next generation display server for Unix-systems.

### High-level design

Computers have multiple **input** and **output** devices, they are responsible for receiving information from you and displaying information to you.

Output devices are generally displays. These resources are shared between all applications, and the role of the **Wayland compositor** is to dispatch input events to the appropriate **Wayland client** and to display their windows in their appropriate place on your outputs.

The process of bringing together all of your application windows for displaying on an output is called **composing**.

### In practice

Multiple distinct software components are part of the graphics/display stack. Some tools of the tools found part of the Linux desktop stack are:

- Mesa for rendering.
- Linux KMS/DRM.
- Buffer allocation with GBM.
- Userspace libdrm library, libinput, evdev, etc.

### The hardware

Interfacing input and output devices is done by several components inside the operating system. This can be interfaces for USB, PCI, etc. Hardware has little concept of what applications are running on the system. The hardware only provides an interface with which it can be commanded to perform work, and do what it is told, regardless of who tells it so. For this reason, only one component is allowed to talk to hardware, this is the **kernel**.

### The kernel

The job of the kernel is to provide an abstraction over the hardware, so that it can be safely accessed from the **userspace**. The userspace is also where the Wayland compositor runs.

The graphics abstraction in the Linux kernel is called **DRM** or **direct rendering manager**[1]. DRM tasks the GPU with work from the userspace. The displays themselves are configured by a subsystem of DRM known as Linux **KMS** or **kernel mode setting**[2].

Input devices are abstracted through an interface called **evdev**[3].

Most kernel interfaces are exposed to the userspace by the way of special files in `/dev`. In the case of DRM, these files are in `/dev/dri`:

- In the form of a primary node for privileged operations like modesetting.
- In the form of render nodes for unprivileged operations like rendering or video decoding.

For evdev, the device nodes are in `/dev/input/event*`.

### The userspace

Applications in the userspace are isolated from the hardware and must work with it via the device nodes provided by the kernel.

[1]https://en.wikipedia.org/wiki/Direct_Rendering_Manager

[2]https://www.kernel.org/doc/html/v4.15/gpu/drm-kms.html

[3]https://en.wikipedia.org/wiki/Evdev

### `libdrm`[4]

`libdrm` is the userspace portion of the DRM subsystem. It's a library providing an C API for interfacing with DRM. `libdrm` is used by Wayland compositors to do mode setting and other DRM operations. It is generally not used by the Wayland clients directly.

### Mesa[5]

Mesa is one of the core parts of the Linux graphics stack. It provides an abstraction over `libdrm` known as **GBM** (Generic Buffer Management) library for allocating buffers on the GPU. It also provides vendor-optimized implementations of OpenGL, etc.

### `libinput`[6]

`libinput` is the userspace abstraction library for evdev. It's responsibility is to receive input events from the kernel, decoding them, and passing them to the Wayland compositor. The Wayland compositor requires special permission to use the evdev files, forcing the Wayland clients to go through the compositor to receive input events (for security reasons).

### `(e)udev`[7]

Dealing with the appearance of new devices from the kernel, as well as configuring permissions for the resulting device nodes in `/dev`, and sending word of the changes to the applications running on the system, is a responsibility that falls onto the userspace. Most systems use `udev` or `eudev` for this purpose. The Wayland compositor uses udev to interface, enumerate and notify about changes with input devices.

### `xkbcommon`[8]

XKB is the original keyboard handling subsystem for Xorg server. Its now an independent keyboard library. Libinput delivers keyboard events in the form of scan codes, which are keyboard dependent. XKB translates the scan codes into generic key "symbols". It also contains a state machine which knows how to process key combinations.

### `pixman`[9]

Library for efficient manipulation of pixel buffers.

### `libwayland`[10]

`libwayland` handles most of the low-level wire protocol.

## The Wayland Package

The Wayland package consists of `libwayland-client`, `libwayland-server`, `wayland-scanner` and `wayland.xml`. When installed they can be found in `/usr/lib & /usr/include`, `/usr/bin` and `/usr/share/wayland/`. This package is the most popular implementation of Wayland.

- `wayland.xml`: Wayland protocols are defined by the XML files.
- `wayland-scanner`: Processes the XML files and generates code from them[11].
- `libwayland`: There are two libraries, one for the client side of the wire protocol and one for the server side.

[4]https://github.com/tobiasjakobi/libdrm
[5]https://mesa3d.org/
[6]https://wayland.freedesktop.org/libinput/doc/latest/
[7]https://en.wikipedia.org/wiki/Udev and https://github.com/eudev-project/eudev
[8]https://xkbcommon.org/
[9]https://www.pixman.org/
[10]https://wayland.freedesktop.org/
[11]Other scanners also exist, like `wayland-rs` and `waymonad-scanner`

## Protocol Design