Building Weston

Most major Linux distributions now include releases of Wayland and Weston in their package management systems.

You can also manually build the stack yourself. The directions below are provided as a guide, but will need adapted for the Linux system you're using. The instructions below assume some familiarity with git, pkg-config, meson, and building and running experimental software.

Wayland is built with Meson. If necessary, the latest Meson can be installed as a user with:

```
$ pip3 install --user meson
```

`pip3` installs meson in `~/.local/` so you might need to add `~/.local/bin` to your `PATH` environment variable, if it is not already added.

Some help for figuring out how to fix dependency problems: Configure fails with "No package 'foo' found" - and how to fix it.

Hardware / Drivers

X output requires DRI2. Output outside of X requires DRM and Kernel Mode Setting (KMS) with the page flip ioctl. These are supported by all open-source drivers.

Setting up the environment

Installing in a custom location

If you do not want to install system wide, you'll need to set the following environment variables to get various libraries to link appropriately:

```
export WLD=$HOME/install   # change this to another location if you prefer
export LD_LIBRARY_PATH=$WLD/lib
export PKG_CONFIG_PATH=$WLD/lib/pkgconfig/:$WLD/share/pkgconfig/
export PATH=$WLD/bin:$PATH
```

Do not set LD_LIBRARY_PATH as your default, it will break things.

You may put the above in a script and source it in the terminal you wish to build the packages.

Installing system wide

To install system wide, you'll need to set WLD differently, add some switches to every meson invocation, and use sudo for the install step. Choose the libdir as appropriate for your distribution and architecture (it may be `/usr/lib32` or `/usr/lib64`).

```
export WLD=/usr
...
meson build/ --prefix=$WLD --libdir=/usr/lib --sysconfdir=/etc
ninja -C build/
sudo ninja -C build/ install
...
```

Wayland libraries

This is required in order to be able to build Mesa with the Wayland EGL platform.

```
$ git clone https://gitlab.freedesktop.org/wayland/wayland.git
$ cd wayland
$ meson build/ --prefix=$WLD
$ ninja -C build/ install
$ cd ..
```

- You can avoid the Doxygen dependency with `-Ddocumentation=false`.
- Under Arch Linux, DocBook dependencies will be needed when building documentation. These include `docbook-xml` and `docbook-xsl`.

Wayland protocols

This contains a set of protocols that add functionality not available in the Wayland core protocol.

```
$ git clone https://gitlab.freedesktop.org/wayland/wayland-protocols.git
$ cd wayland-protocols
$ meson build/ --prefix=$WLD
$ ninja -C build/ install
$ cd ..
```

Mesa

Mesa EGL and Mesa Vulkan stacks support Wayland. Weston's hardware acceleration (GL-renderer) depends on EGL GBM platform. Many Wayland applications, including some Weston demos, depend on EGL Wayland platform.

For building Mesa, refer to the upstream build instructions. When configuring Mesa to make the most out of Weston, make sure that OpenGL ES 2 and GBM are enabled, and that EGL platforms includes wayland.

If you plan to compile XWayland you may want to install any dependencies it needs now. This is so Mesa uses the same header files as xwayland.

libinput

Libinput translates evdev events into Wayland events. It has been split from Weston as the code is reusable by any Wayland compositor on Linux. Head to libinput docs for instructions on how to build it.

Weston and demo applications

Weston is the reference implementation of a Wayland compositor. It's available in the weston repo and comes with a few demo applications.

Weston's Meson build does not do autodetection and it defaults to all features enabled, which means you likely hit missing dependencies on the first try. If a dependency is avoidable through a build option, the error message should tell you what option can be used to avoid it. You may need to disable several features if you want to avoid certain dependencies.

```
$ git clone https://gitlab.freedesktop.org/wayland/weston.git
$ cd weston
$ meson build/ --prefix=$WLD
$ ninja -C build/ install
$ cd ..
```

The `meson` command populates the build directory. This step can fail due to missing dependencies. Any build options you want can be added on that line, e.g. `meson build/ --prefix=$WLD -Dsimple-dmabuf-drm=intel`. All the build options can be found in the file `meson_options.txt`.

Once the build directory has been successfully populated, you can inspect the configuration with `meson configure build/`. If you need to change an option, you can do e.g. `meson configure build/ -Dsimple-dmabuf-drm=intel`.

Running Weston

Weston creates its unix socket file (for example, `wayland-0`) in the directory specified by the required environment variable `$XDG_RUNTIME_DIR`. Clients use the same variable to find that socket. This is provided using systemd by some distributions (Fedora, Arch since June 2012 or Exherbo). Ubuntu began providing it in Quantal. It is not provided by Gentoo.

If `$XDG_RUNTIME_DIR` isn't automatically set for you, you can setup rundird or pam_rundir.

Optionally, copy the `weston.ini` file and edit it to set a background image that you like:

```
$ mkdir -p ~/.config
$ cp weston/weston.ini ~/.config
$ $EDITOR ~/.config/weston.ini
```

If `$DISPLAY` is set, then Weston will run under X in a window and take input from X. Run the compositor by typing:

```
$ weston
```

Otherwise (outside of X) it will run on the KMS/DRM framebuffer and take input from evdev devices.

If logind is not available, running Weston directly won't work. A privileged helper, weston-launch, will be required in this case. It needs to be setuid-root.

To run clients, the second button in the top bar will run weston-terminal, from which you can run clients. It is also possible to run clients from a different VT when running on hardware, or from an xterm if running under X. There are a few demo clients available in the weston build directory, but they are all pretty simple and mostly for testing specific features in the wayland protocol:

- 'weston-terminal' is a simple terminal emulator, not very compliant at all, but works well enough for bash
- 'weston-flower' draws a flower on the screen, testing the frame protocol
- 'weston-gears' glxgears, but for wayland
- 'weston-smoke' tests SHM buffer sharing
- 'weston-image' loads the image files passed on the command line and shows them
- 'weston-view' does the same for pdf files
- 'weston-resizor' demonstrates smooth window resizing (use up and down keys)
- 'weston-eventdemo' reports libtoytoolkit's events to console (see weston-eventdemo --help)

Optional environment variables which will get you more debugging output:

```
export MESA_DEBUG=1
export EGL_LOG_LEVEL=debug
export LIBGL_DEBUG=verbose
export WAYLAND_DEBUG=1
```

XWayland

Directions for building support for X clients (XWayland)