

Activation script

The activation script is a bash script called to activate the new configuration which resides in a NixOS system in `$out/activate`. Since its contents depend on your system configuration, the contents may differ. This chapter explains how the script works in general and some common NixOS snippets. Please be aware that the script is executed on every boot and system switch, so tasks that can be performed in other places should be performed there (for example letting a directory of a service be created by systemd using mechanisms like `StateDirectory`, `CacheDirectory`, ... or if that's not possible using `preStart` of the service).

Activation scripts are defined as snippets using . They can either be a simple multiline string or an attribute set that can depend on other snippets. The builder for the activation script will take these dependencies into account and order the snippets accordingly. As a simple example:

```
system.activationScripts.my-activation-script = {
  deps = [ "etc" ];
  # supportsDryActivation = true;
  text = ''
    echo "Hallo i bims"
  '';
};
```

This example creates an activation script snippet that is run after the `etc` snippet. The special variable `supportsDryActivation` can be set so the snippet is also run when `nixos-rebuild dry-activate` is run. To differentiate between real and dry activation, the `$NIXOS_ACTION` environment variable can be read which is set to `dry-activate` when a dry activation is done.

An activation script can write to special files instructing `switch-to-configuration` to restart/reload units. The script will take these requests into account and will incorporate the unit configuration as described above. This means that the activation script will “fake” a modified unit file and `switch-to-configuration` will act accordingly. By doing so, configuration like `systemd.services.<name>.restartIfChanged` is respected. Since the activation script is run **after** services are already stopped, `systemd.services.<name>.stopIfChanged` cannot be taken into account anymore and the unit is always restarted instead of being stopped and started afterwards.

The files that can be written to are `/run/nixos/activation-restart-list` and `/run/nixos/activation-reload-list` with their respective counterparts for dry activation being `/run/nixos/dry-activation-restart-list` and `/run/nixos/dry-activation-reload-list`. Those files can contain newline-separated lists of unit names where duplicates are being ignored. These files are not created automatically and activation scripts must take the possibility into account that they have to create them first.

NixOS snippets

There are some snippets NixOS enables by default because disabling them would most likely break your system. This section lists a few of them and what they do:

- `binsh` creates `/bin/sh` which points to the runtime shell
- `etc` sets up the contents of `/etc`, this includes systemd units and excludes `/etc/passwd`, `/etc/group`, and `/etc/shadow` (which are managed by the `users` snippet)
- `hostname` sets the system's hostname in the kernel (not in `/etc`)
- `modprobe` sets the path to the `modprobe` binary for module auto-loading
- `nix` prepares the nix store and adds a default initial channel
- `specialfs` is responsible for mounting filesystems like `/proc` and `sys`
- `users` creates and removes users and groups by managing `/etc/passwd`, `/etc/group` and `/etc/shadow`. This also creates home directories
- `usrbinenv` creates `/usr/bin/env`
- `var` creates some directories in `/var` that are not service-specific
- `wrappers` creates setuid wrappers like `sudo`



/Plugins/Advanced

Advanced

This page documents a few advanced things about the Hyprland Plugin API.

- Using Function Hooks
 - Member functions
 - Why use `findFunctionsByName`?
- Using the config
- Further

Using Function Hooks

Important

Function hooks are only available on `AMD64` (`x86_64`). Attempting to hook on any other arch will make Hyprland simply ignore your hooking attempt.

Function hooks are intimidating at first, but when used properly can be *extremely* powerful.

Function hooks allow you to intercept any call to the function you hook.

Let's look at a simple example:

```
void Events::listener_monitorFrame(void* owner, void* data)
```

will be the function we want to hook. `Events::` is a namespace, not a class, so this is just a plain function.

```
// make a global instance of a hook class for this hook
inline CFunctionHook* g_pMonitorFrameHook = nullptr;
// create a pointer typedef for the function we are hooking.
typedef void (*origMonitorFrame)(void*, void*);
```

```
// our hook
void hkMonitorFrame(void* owner, void* data) {
    (*(origMonitorFrame)g_pMonitorFrameHook->m_pOriginal)(owner, data);
}

APICALL EXPORT PLUGIN_DESCRIPTION_INFO PLUGIN_INIT(HANDLE handle) {
    // stuff...

    // create the hook
    static const auto METHODS = HyprlandAPI::findFunctionsByName(PHANDLE,
"listener_monitorFrame");
    g_pMonitorFrameHook = HyprlandAPI::createFunctionHook(handle,
METHODS[0].address, (void*)&hkMonitorFrame);

    // init the hook
    g_pMonitorFrameHook->hook();

    // further stuff...
}
```

We have just made a hook. Now, whenever Hyprland calls
Events::`listener_monitorFrame`, our hook will be called instead!

This way, you can run code before / after the function, modify the inputs or results, or even block the function from executing.

`CFunctionHook` can also be unhooked whenever you please. Just run `unhook()`. It can be rehooked later by calling `hook()` again.

Member functions

For members, e.g. `CCompositor::focusWindow(CWindow*, wlr_surface*)` you will also need to add the `thisptr` argument to your hook:

```
typedef void (*origFocusWindow)(void*, CWindow*, wlr_surface*);

void hkFocusWindow(void* thisptr, CWindow* pWindow, wlr_surface* pSurface) {
    // stuff...

    // and if you want to call the original...
    (*(origFocusWindow)g_pFocusWindowHook->m_pOriginal)(thisptr, pwindow,
pSurface);
}

APICALL EXPORT PLUGIN_DESCRIPTION_INFO PLUGIN_INIT(HANDLE handle) {
```

```
// stuff...

    static const auto METHODS = HyprlandAPI::findFunctionsByName(PHANDLE,
"focusWindow");
    g_pFocusWindowHook = HyprlandAPI::createFunctionHook(handle,
METHODS[0].address, (void*)&hkFocusWindow);
    g_pFocusWindowHook->hook();

    // further stuff...
}
```

Warning

Please note method lookups are slow and should not be used often. The entries *will not* change during runtime, so it's a good idea to make the lookups `static`.

Why use `findFunctionsByName`?

Why use that instead of e.g. `&CCompositor::focusWindow`? Two reasons:

- 1 - less breakage. Whenever someone updates hyprland, that address might become invalid. `findFunctionsByName` is more resilient. As long as the function exists, it will be found.
- 2 - error handling. The method array contains, besides the address, the signatures. You can verify those to make 100% sure you got the right function, or throw an error if it was not found.

Using the config

You can register config values in the `PLUGIN_INIT` function:

```
APICALL EXPORT PLUGIN_DESCRIPTION_INFO PLUGIN_INIT(HANDLE handle) {
    // stuff...

    HyprlandAPI::addConfigValue(PHANDLE, "plugin:example:exampleInt",
SConfigValue{.intValue = 1});

    // further stuff...
}
```

Plugin variables **must** be in the `plugins:` category. Further categories are up to you. It's generally a good idea to group all variables from your plugin in a subcategory with the plugin name, e.g. `plugins:myPlugin:variable1`.

For retrieving the values, call `HyprlandAPI::getConfigValue` .

Please remember that the pointer to your config value will never change after `PLUGIN_INIT` , so to greatly optimize performance, make it static:

```
static auto* const MYVAR      = &HyprlandAPI::getConfigValue(PHANDLE,  
"plugin:myPlugin:variable1")->intValue;
```

Further

Read the API at `src/plugins/PluginAPI.hpp` , check out the official plugins.

And, most importantly, have fun!

The Hyprland Wiki, built with Hugo.



/Configuring/Animations

Animations

Table of contents

- Table of contents
- General
 - Examples
 - Animation tree
- Curves
 - Example
- Extras

General

Animations are declared with the `animation` keyword.

```
animation=NAME, ONOFF, SPEED, CURVE, STYLE  
or  
animation=NAME, ONOFF, SPEED, CURVE
```

`ONOFF` can be either 0 or 1, 0 to disable, 1 to enable. *note:* if it's 0, you can omit further args.

`SPEED` is the amount of ds (1ds = 100ms) the animation will take

`CURVE` is the bezier curve name, see curves.

`STYLE` (optional) is the animation style

The animations are a tree. If an animation is unset, it will inherit its parent's values. See the animation tree.

Examples

```
animation=workspaces,1,8,default
animation=windows,1,10,myepiccurve,slide
animation=fade,0
```

Animation tree

```
global
↳ windows - styles: slide, popin
    ↳ windowsIn - window open
    ↳ windowsOut - window close
    ↳ windowsMove - everything in between, moving, dragging, resizing.
↳ fade
    ↳ fadeIn - fade in (open) -> layers and windows
    ↳ fadeOut - fade out (close) -> layers and windows
    ↳ fadeSwitch - fade on changing activewindow and its opacity
    ↳ fadeShadow - fade on changing activewindow for shadows
    ↳ fadeDim - the easing of the dimming of inactive windows
↳ border - for animating the border's color switch speed
↳ borderangle - for animating the border's gradient angle - styles: once
(default), loop
↳ workspaces - styles: slide, slidevert, fade, slidefade, slidefadevert
    ↳ specialWorkspace - styles: same as workspaces
```

Curves

Defining your own Bezier curve can be done with the `bezier` keyword:

```
bezier=NAME,X0,Y0,X1,Y1
```

where `NAME` is the name, and the rest are two points for the Cubic Bezier. A good website to design your bezier can be found here, on cssportal.com, but if you want to instead choose from a list of beziers, you can check out easings.net.

Example

```
bezier=overshot,0.05,0.9,0.1,1.1
```

Extras

For animation style `popin` in `windows`, you can specify a minimum percentage to start from. For example, the following will make the animation 80% -> 100% of the size:

```
animation=windows,1,8,default,popin 80%
```

For animation styles `slidefade` and `slidefadevert` in `workspaces`, you can specify a movement percentage. For example, the following will make windows move 20% of the screen width:

```
animation=workspaces,1,8,default,slidefade 20%
```

The Hyprland Wiki, built with Hugo.



[/Useful Utilities](#) / App Clients

App Clients

Some clients are known for being a massive pain under Wayland. Here are great replacements for them:

Discord

- WebCord is a Discord client based on the latest Electron, with support for Wayland Ozone platform, as well as PipeWire screensharing. It has tons of great features and tries not to infringe on the Discord ToS.
- gtkcord4 is a Discord client written in GTK4. While it does infringe on Discord's ToS, it's relatively safe and doesn't rely on any webview technologies.

Matrix/Element

Fractal is a Matrix client written in GTK4. Much like Discord, Element is known to have a lot of problems as a result of being based on Electron. Fractal currently doesn't support VoIP calling, but all other features are supported, including E2EE and cross-device verification.

The Hyprland Wiki, built with Hugo.



 /Useful Utilities/App Launchers

App Launchers

Wofi

Wofi is a GTK-based customizable launcher for wayland. [SourceHut](#).

Rofi (Wayland fork)

Rofi, but with Wayland support. [GitHub](#).

bemenu

bemenu is a Wayland-native replacement for dmenu. [GitHub](#).

fuzzel

Fuzzel is an application launcher for wlroots based Wayland compositors, similar to rofi's drun mode. [Codeberg](#)

tofi

tofi is an extremely fast and simple yet highly customizable dmenu / rofi replacement for wlroots-based Wayland compositors. When configured correctly, tofi can get on screen within a single frame. [Github](#)

For a more comprehensive list of launchers, check [awesome-hyprland](#).

The Hyprland Wiki, built with Hugo.

Hyprland Wiki

 /Configuring/Binds

Binds

Table of Contents

- Table of Contents
- Basic
 - Uncommon syms / binding with a keycode
- Misc
 - Unbind
 - Mouse buttons
 - Only modkeys
 - Mouse wheel
 - Switches
 - Multiple binds to one key
- Bind flags
- Mouse Binds
- Binding mods
- Global Keybinds
 - Classic
 - DBus Global Shortcuts
- Submaps

Basic

```
bind=MODS, key, dispatcher, params
```

for example,

```
bind=SUPER_SHIFT,Q,exec,firefox
```

will bind opening firefox to SUPER + SHIFT + Q

Tip

For binding keys without a modkey, leave it empty:

```
bind=,Print,exec,grim
```

For a complete mod list, see *Variables*.

The dispatcher list can be found in *Dispatchers*.

Uncommon syms / binding with a keycode

See the xkbcommon-keysyms.h header for all the keysyms. The name you should use is the segment after `XKB_KEY_`.

If you are unsure of what your key's name is, you can use `xev` or `wew` to find that information.

If you want to bind by a keycode, you can just input it in the KEY position with a `code` prefix, e.g.:

```
bind=SUPER,code:28,exec,amongus
```

Will bind SUPER + T. (T is keycode 28.) - You can also use `xev` or `wew` to find keycodes.

Misc

Unbind

You can also unbind with `unbind`, e.g.:

```
unbind=SUPER,0
```

May be useful for dynamic keybindings with `hyprctl`.

```
hyprctl keyword unbind SUPER,0
```

Mouse buttons

You can also bind mouse buttons, by prefacing the mouse keycode with `mouse:`, for example:

```
bind=SUPER,mouse:272,exec,amongus
```

will bind it to SUPER + LMB.

Only modkeys

For binding only modkeys, you need to use the TARGET modmask (with the activating mod) and the `r` flag, e.g.:

```
bindr=SUPERALT,Alt_L,exec,amongus
```

Mouse wheel

You can also bind the mouse wheel with `mouse_up` and `mouse_down` (or `mouse_left` and `mouse_right` if your wheel supports horizontal scrolling):

```
bind=SUPER,mouse_down,workspace,e-1
```

(control the reset time with `binds:scroll_event_delay`)

Switches

Useful for binding e.g. the lid close/open event:

```
# trigger when the switch is toggled
bindl=,switch:[switch name],exec,swaylock
# trigger when the switch is turning on
bindl=,switch:on:[switch name],exec,hyprctl keyword monitor "eDP-1, 2560x1600, 0x0, 1"
# trigger when the switch is turning off
bindl=,switch:off:[switch name],exec,hyprctl keyword monitor "eDP-1, disable"
```

check out your switches in `hyprctl devices`.

Multiple binds to one key

You can trigger multiple actions with one keybind by assigning multiple binds to one combination, e.g.:

```
# to switch between windows in a floating workspace
bind = SUPER,Tab,cyclenext,          # change focus to another window
bind = SUPER,Tab,bringactivetotop,    # bring it to the top
```

The keybinds will be executed in the order they were created. (top to bottom)

Bind flags

`bind` supports flags in this format:

```
bind[flags]=...
```

e.g.:

```
bindrl=MOD,KEY,exec,amongus
```

Flags:

```
l -> locked, aka. works also when an input inhibitor (e.g. a lockscreen) is active.
r -> release, will trigger on release of a key.
e -> repeat, will repeat when held.
n -> non-consuming, key/mouse events will be passed to the active window in addition to triggering the dispatcher.
m -> mouse, see below
t -> transparent, cannot be shadowed by other binds.
i -> ignore mods, will ignore modifiers.
```

Example Usage:

```
# Example volume button that allows press and hold, volume limited to 150%
bind=e, XF86AudioRaiseVolume, exec, wpctl set-volume -l 1.5
@DEFAULT_AUDIO_SINK@ 5%+
```

```
# Example volume button that will activate even while an input inhibitor is active  
bindl=, XF86AudioLowerVolume, exec, wpctl set-volume @DEFAULT_AUDIO_SINK@ 5%-  
  
# Start wofi opens wofi on first press, closes it on second  
bindr=SUPER, SUPER_L, exec, pkill wofi || wofi  
  
# See Mouse Binds section for bindm usage
```

Mouse Binds

Mouse binds are binds that heavily rely on a mouse, usually its movement. They will have one less arg, and look for example like this:

```
bindm=ALT, mouse:272, movewindow
```

this will create a bind with ALT + LMB to move the window with your mouse.

Available mouse binds:

| Name | Description | Params |
|--------------|---------------------------|--|
| movewindow | moves the active window | none |
| resizewindow | resizes the active window | 1 - resize and keep window aspect ratio, 2 - resize and ignore keepaspectratio window rule/prop, none or anything else for normal resize |

Common mouse buttons' codes:

```
LMB -> 272  
RMB -> 273
```

for more, you can of course use `wev` to check.

Tip

Mouse binds, despite their name, behave like normal binds. You are free to use whatever keys / mods you please. When held, the mouse function will be activated.

Binding mods

You can bind a mod alone like this:

```
bindr=ALT,Alt_L,exec,amongus
```

Global Keybinds

Classic

Yes, you heard this right, Hyprland does support global keybinds for ALL apps, including OBS, Discord, Firefox, etc.

See the `pass` dispatcher for keybinds.

Let's take OBS as an example: the "Start/Stop Recording" keybind is set to `SUPER + F10`, and you want to make it work globally.

Simply add

```
bind = SUPER,F10,pass,^(com\obsproject\Studio)$
```

to your config and you're done.

`pass` will pass the PRESS and RELEASE events by itself, no need for a `bindr`. This also means that push-to-talk will work flawlessly with one `pass`, e.g.:

```
bind=,mouse:276,pass,^(TeamSpeak 3)$
```

Will pass MOUSE5 to TeamSpeak3.

Important

XWayland is a bit wonky. Make sure that what you're passing is a "global Xorg keybind", otherwise passing from a different XWayland app may not work.

It works flawlessly with all native Wayland applications though.

DBus Global Shortcuts

Some applications may already support the GlobalShortcuts portal in xdg-desktop-portal.

If that's the case, then it's recommended to use this method instead of `pass`.

Open your desired app and issue `hyprctl globalshortcuts`. This will give you a list of currently registered shortcuts with their description(s).

Choose whichever you like, for example `coolApp:myToggle`

Bind it to whatever you want with the `global` dispatcher:

```
bind = SUPERSHIFT, A, global, coolApp:myToggle
```

Tip

Please note that this function will *only* work with XDPH.

Submaps

If you want keybind submaps, also known as *modes* or *groups*, for example if you press `ALT + R`, you can enter a “resize” mode, resize with arrow keys, and leave with escape, do it like this:

```
# will switch to a submap called resize
bind=ALT,R,submap,resize

# will start a submap called "resize"
submap=resize

# sets repeatable binds for resizing the active window
binde=,right,resizeactive,10 0
binde=,left,resizeactive,-10 0
binde=,up,resizeactive,0 -10
binde=,down,resizeactive,0 10

# use reset to go back to the global submap
bind=,escape,submap,reset
```

```
# will reset the submap, meaning end the current one and return to the global  
one  
submap=reset  
  
# keybinds further down will be global again...
```

IMPORTANT: do not forget a keybind to reset the keymap while inside it! (In this case, escape)

If you get stuck inside a keymap, you can use `hyprctl dispatch submap reset` to go back. If you do not have a terminal open, tough luck buddy. You have been warned.

You can also set the same keybind to perform multiple actions, such as resize and close the submap, like so:

```
bind=ALT,R,submap,resize  
  
submap=resize  
  
bind=,right,resizeactive,10 0  
bind=,right,submap,reset  
# ...  
  
submap=reset
```

This works because the binds are executed in the order they appear, and assigning multiple actions per bind is possible.

The Hyprland Wiki, built with Hugo.

Hyprland Wiki

 /Nix/Cachix

Cachix

NOTE: This page only applies to the flake package. You can safely skip this if you use the Nixpkgs package.

The Hyprland flake is not built by Hydra, so it is not cached in cache.nixos.org, like the rest of Nixpkgs.

Instead of requiring you to build Hyprland (and its dependencies, which may include `mesa`, `ffmpeg`, etc), we provide a Cachix cache that you can add to your Nix configuration.

The Hyprland Cachix exists to cache the `hyprland` packages and any dependencies not found in cache.nixos.org.

Note

In order for Nix to take advantage of the cache, it has to be enabled **before** using the Hyprland flake package.

```
# configuration.nix
{
  nix.settings = {
    substituters = ["https://hyprland.cachix.org"];
    trusted-public-keys = ["hyprland.cachix.org-
1:a7pgxzMz7+chwVL3/pzj6jIBMioiJM7ypFP8PwtkuGc="];
  };
}
```

Important

Do **not** override Hyprland's `nixpkgs` input unless you know what you are doing.

Doing so will make the cache useless, since you're building from a different Nixpkgs commit.

The Hyprland Wiki, built with Hugo.



/Useful Utilities/Clipboard Managers

Clipboard Managers

Starting method: manual (exec-once)

Clipboard Managers are useful tools that allow one to manage their copied items, be-it texts or images.

Some common ones used are `copyq`, `clipman` and `cliphist`.

`clipman` - Utilizes Wayland with `wl-clipboard` support and stores text only [Github](#)

`cliphist` - Utilizes Wayland with `wl-clipboard` and can store both images and text [Github](#)

`wl-clip-persist` - When we copy something on Wayland (using `wl-clipboard`) and close the application we copied from, the copied data disappears from the clipboard and we cannot paste it anymore. So to fix this problem we can use a program called as `wl-clip-persist` which will preserve the data in the clipboard after the application is closed. [Github](#)

copyq

Start by adding the following lines to your `~/.config/hypr/hyprland.conf`

```
exec-once = copyq --start-server
```

If your `copyq`'s main window cannot close/hide properly, try to enable its “Hide main window” option in Layout configuration tab in Preferences dialog.

cliphist

Start by adding the following lines to your `~/.config/hypr/hyprland.conf`

```
exec-once = wl-paste --type text --watch cliphist store #Stores only text data
```

```
exec-once = wl-paste --type image --watch cliphist store #Stores only image data
```

Do note that any of the above lines can be disabled based on your needs

To bind `cliphist` to a hotkey and display it under `rofi` or `dmenu` or `wofi`, again head over to `~/.config/hypr/hyprland.conf`

For rofi users

```
bind = SUPER, V, exec, cliphist list | rofi -dmenu | cliphist decode | wl-copy
```

For dmenu users

```
bind = SUPER, V, exec, cliphist list | dmenu | cliphist decode | wl-copy
```

For wofi users

```
bind = SUPER, V, exec, cliphist list | wofi --dmenu | cliphist decode | wl-copy
```

The binds mention above correspond to SUPER+V to access the clipboard history

For further info, please refer to the repository mentioned above

clipman

Start by adding the following line to your `~/.config/hypr/hyprland.conf`

```
exec-once = wl-paste -t text --watch clipman store --no-persist
```

If you wish to use it as a primary clipboard manager, use this instead

```
exec-once = wl-paste -p -t text --watch clipman store -P --histpath="~/local/share/clipman-primary.json"
```

And also make sure to create a file named `clipman-primary.json` in `~/local/share/clipman-primary.json`

Now bind the `clipman` like this:

For rofi users

```
bind = SUPER, V, exec, clipman pick -t rofi
```

For dmenu users

```
bind = SUPER, V, exec, clipman pick -t dmenu
```

For wofi users

```
bind = SUPER, V, exec, clipman pick -t wofi
```

So on and so forth. For further information, please refer to the repository mentioned above

The Hyprland Wiki, built with Hugo.



/Configuring/Configuring Hyprland

Configuring Hyprland

The config is located in `~/.config/hypr/hyprland.conf`.

Hyprland will automatically generate an example config for you if you don't have one. You can find an example config [here](#).

By removing the line `autogenerated=1` you'll remove the yellow warning.

There is no "reload" keybind. The config is reloaded the moment you save it.

Start a section with `name {` and end in `}` ***in separate lines!***

Important

The default config is not complete and does not list all the options / features of Hyprland. Please refer to this wiki page and the pages linked further down here for full configuration instructions.

Make sure to read the Variables page as well. It covers all the toggleable / numerical options.

Line style

Every config line is a command followed by a value.

`COMMAND=VALUE`

The command can be a variable, or a special keyword (described further in this page)

You are **allowed to** input trailing spaces at the beginning and end.

e.g.:

COMMAND = VALUE

is valid.

Comments

Comments are started with the `#` character.

If you want to escape it (put an actual `#` and not start a comment) you can use `##`. It will be turned into a single `#` that WILL be a part of your line.

Basic configuring

To configure the “options” of Hyprland, animations, styling, etc. see Variables.

Advanced configuring

Some keywords (binds, curves, execs, monitors, etc.) are not variables but define special behavior.

See all of them in Keywords and the sidebar.

The Hyprland Wiki, built with Hugo.



/Crashes and Bugs

Crashes and Bugs

Getting the log

If you are in a TTY, and the hyprland session that crashed was the last one you launched, the log will be printed with

```
cat /tmp/hypr/$(ls -t /tmp/hypr/ | head -n 1)/hyprland.log
```

feel free to save it to a file, save, copy, etc.

if you are in a Hyprland session, and you want the log of the last session, use

```
cat /tmp/hypr/$(ls -t /tmp/hypr/ | head -n 2 | tail -n 1)/hyprland.log
```

Crashes at launch

Diagnose the issue by what is in the log:

- `sWLRBackend` was `NUL!` -> launch in the TTY and refer to the wlr logs in RED.
- Monitor X has NO PREFERRED MODE, and an INVALID one was requested -> your monitor is bork.
- Other -> see the coredump. Use `coredumpctl`, find the latest one's PID and do `coredumpctl info PID`.
- failing on a driver (e.g. `radeon`) -> try compiling with `make legacyrenderer`, if that doesn't help, report an issue.
- failing on `wlr-xxx` -> try compiling with `make legacyrenderer`, if that doesn't help, report an issue, and also refer to the TTY wlr logs in RED like in the first point.
- failing on `Hyprland` -> report an issue.

Crashes not at launch

Report an issue on GitHub or on the Discord server.

Bugs

First of all, ***READ THE FAQ PAGE***

If your bug is not listed there, you can ask on the Discord server or open an issue on GitHub.

Building the Wayland stack with ASan

If requested, this is the deepest level of memory issue debugging possible.

Prepare yourself mentally, and then:

recommended to do in tty

```
clone wayland ( git clone --recursive https://gitlab.freedesktop.org/wayland/wayland ) clone hyprland ( git clone --recursive https://github.com/hyprwm/Hyprland )
```

add these envs to your Hyprland config to reset ASAN_OPTIONS for children and set LD_PRELOAD:

```
env = ASAN_OPTIONS,detect_odr_violation=0  
env = LD_PRELOAD,/usr/lib/libasan.so.8.0.0
```

Please note to check the asan .so version on your system with ls /usr/lib | grep libasan

wayland:

```
meson ./build --prefix=/usr --buildtype=debug -Db_sanitize=address  
sudo ninja -C build install
```

The Wayland build will likely fail citing missing dependencies such as Doxygen, these dependencies will likely be available from your distros package manager.

hyprland:

```
cmake --no-warn-unused-cli -DCMAKE_BUILD_TYPE:STRING=Debug  
-DWITH_ASAN:STRING=True -S . -B ./build -G Ninja  
cmake --build ./build --config Debug --target all -j`nproc 2>/dev/null ||  
getconf NPROCESSORS_CONF`  
cd ./subprojects/wlroots  
rm -rf ./build  
meson ./build --prefix=/usr --buildtype=debug -Db_sanitize=address  
ninja -C build  
cd ../../..  
sudo make install
```

Exit Hyprland to a TTY, cd to the cloned hyprland, and launch it:

```
ASAN_OPTIONS="detect_odr_violation=0,log_path=asan.log" ./build/Hyprland  
-c ~/.config/hypr/hyprland.conf
```

open your terminal

Do whatever you used to do in order to crash the compositor.

Please note many apps will refuse to launch. Notably complex applications, like e.g. browsers.

Once it crashes, go to ~ or cwd and look for asan.log.XXXXX files. Zip all and attach to the issue.

once you are done, to revert your horribleness of no app opening without the ld preload just go to the cloned wayland and do

```
sudo rm -rf ./build  
meson ./build --prefix=/usr --buildtype=release  
sudo ninja -C build install
```

To revert the changes to hyprland and wlroots, do inside the cloned hyprland:

```
make all && sudo make install
```

The Hyprland Wiki, built with Hugo.



/Configuring/Dispatchers

Dispatchers

Table of contents

- Table of contents
- Parameter explanation
- List of Dispatchers
 - Grouped (tabbed) windows
- Workspaces
- Special Workspace
- Executing with rules

Please keep in mind some layout-specific dispatchers will be listed in the layout pages (See the sidebar).

Parameter explanation

| Param type | Description |
|--------------|---|
| window | a window. Any of the following: Class regex, title: and a title regex, pid: and the pid, address: and the address, floating , tiled |
| workspace | see below. |
| direction | l r u d left right up down |
| monitor | One of: direction, ID, name, current , relative (e.g. +1 or -1) |
| resizeparams | relative pixel delta vec2 (e.g. 10 -10), optionally a percentage of the |

| Param type | Description |
|------------|--|
| | window size (e.g. 20 25%) or exact followed by an exact vec2 (e.g. exact 1280 720), optionally a percentage of the screen size (e.g. exact 50% 50%) |
| floatvalue | a relative float delta (e.g -0.2 or +0.2) or exact followed by a the exact float value (e.g. exact 0.5) |
| zheight | top or bottom |

List of Dispatchers

| Dispatcher | Description | Params |
|-------------|--|-------------------------------------|
| exec | executes a shell command | command (supports rules, see below) |
| execr | executes a raw shell command (does not support rules) | command |
| pass | passes the key (with mods) to a specified window. Can be used as a workaround to global keybinds not working on Wayland. | window |
| killactive | closes (not kills) the active window | none |
| closewindow | closes a specified window | window |
| workspace | changes the workspace | workspace |

| Dispatcher | Description | Params |
|-----------------------|---|---|
| movetoworkspace | moves the focused window to a workspace | workspace OR workspace,window for a specific window |
| movetoworkspacesilent | same as above, but doesn't switch to the workspace | workspace OR workspace,window for a specific window |
| togglefloating | toggles the current window's floating state | left empty / active for current, or window for a specific window |
| fullscreen | toggles the focused window's fullscreen state | 0 - fullscreen (takes your entire screen), 1 - maximize (keeps gaps and bar(s)) |
| fakefullscreen | toggles the focused window's internal fullscreen state without altering the geometry | none |
| dpms | sets all monitors' DPMS status. Do not use with a keybind directly. | on , off , or toggle . For specific monitor add monitor name after a space |
| pin | pins a window (i.e. show it on all workspaces) <i>note: floating only</i> | left empty / active for current, or window for a specific window |
| movefocus | moves the focus in a direction | direction |
| movewindow | moves the active window in a direction or to a monitor. For floating windows, moves the | direction or mon: and a monitor |

| Dispatcher | Description | Params |
|-------------------|--|--|
| | window to the screen edge in that direction | |
| swapwindow | swaps the active window with another window in the given direction | direction |
| centerwindow | center the active window <i>note: floating only</i> | none (for monitor center) or 1 (to respect monitor reserved area) |
| resizeactive | resizes the active window | resizeparams |
| moveactive | moves the active window | resizeparams |
| resizewindowpixel | resizes a selected window | resizeparams, window , e.g. 100 100,^(kitty)\$ |
| movewindowpixel | moves a selected window | resizeparams, window |
| cyclenext | focuses the next window on a workspace | none (for next) or prev (for previous) additionally tiled for only tiled, floating for only floating. prev tiled is ok. |
| swapnext | swaps the focused window with the next window on a workspace | none (for next) or prev (for previous) |
| focuswindow | focuses the first window matching | window |

| Dispatcher | Description | Params |
|--------------------------------|--|--|
| focusmonitor | focuses a monitor | monitor |
| splitratio | changes the split ratio | floatvalue |
| toggleopaque | toggles the current window to always be opaque. Will override the opaque window rules. | none |
| movecursortocorner | moves the cursor to the corner of the active window | direction, 0 - 3, bottom left - 0, bottom right - 1, top right - 2, top left - 3 |
| movecursor | moves the cursor to a specified position | x y |
| renameworkspace | rename a workspace | id name , e.g. 2 work |
| exit | exits the compositor with no questions asked. | none |
| forcerendererreload | forces the renderer to reload all resources and outputs | none |
| movecurrentworkspacetomonitor | Moves the active workspace to a monitor | monitor |
| focusworkspaceoncurrentmonitor | Focuses the requested workspace on the current monitor, swapping the current workspace to a different monitor if necessary. If you want XMonad/Qtile-style workspace | workspace |

| Dispatcher | Description | Params |
|------------------------|--|--|
| | switching, replace workspace in your config with this. | |
| moveworkspacetomonitor | Moves a workspace to a monitor | workspace and a monitor separated by a space |
| swapactiveworkspaces | Swaps the active workspaces between two monitors | two monitors separated by a space |
| bringactivetotop | <i>Deprecated</i> in favor of alterzorder. Brings the current window to the top of the stack | none |
| alterzorder | Modify the window stack order of the active or specified window. Note: this cannot be used to move a floating window behind a tiled one. | zheight[,window] |
| togglespecialworkspace | toggles a special workspace on/off | none (for the first) or name for named (name has to be a special workspace's name) |
| focusurgentorlast | Focuses the urgent window or the last window | none |
| togglegroup | toggles the current active window into a group | none |

| Dispatcher | Description | Params |
|--------------------|--|---|
| changegroupactive | switches to the next window in a group. | b - back, f - forward, or index start at 1 |
| focuscurrentorlast | Switch focus from current to previously focused window | none |
| lockgroups | Locks the groups (all groups will not accept new windows) | lock for locking, unlock for unlocking, toggle for toggle |
| lockactivegroup | Lock the focused group (the current group will not accept new windows or be moved to other groups) | lock for locking, unlock for unlocking, toggle for toggle |
| moveintogroup | Moves the active window into a group in a specified direction. No-op if there is no group in the specified direction. | direction |
| moveoutofgroup | Moves the active window out of a group. No-op if not in a group | none |
| movewindoworgroup | Behaves as moveintogroup if there is a group in the given direction. Behaves as moveoutofgroup if there is no group in the given direction relative to the active group. Otherwise behaves like movewindow . | direction |

| Dispatcher | Description | Params |
|---------------------|--|---------------------------------------|
| movegroupwindow | Swaps the active window with the next or previous in a group | b for back, anything else for forward |
| denywindowfromgroup | Prohibit the active window from becoming or being inserted into group | on , off or, toggle |
| setignoregrouplock | Temporarily enable or disable binds:ignore_group_lock | on , off , or toggle |
| global | Executes a Global Shortcut using the GlobalShortcuts portal. See here | name |
| submap | Change the current mapping group. See Submaps | reset or name |

Warning

it is NOT recommended to set DPMS with a keybind directly, as it might cause undefined behavior. Instead, consider something like

```
bind = MOD,KEY,exec,sleep 1 && hyprctl dispatch dpms off
```

Grouped (tabbed) windows

Hyprland allows you to make a group from the current active window with the `togglegroup` bind dispatcher.

A group is like i3wm's "tabbed" container. It takes the space of one window, and you can change the window to the next one in the tabbed "group" with the `changegroupactive` bind dispatcher.

The new group's border colors are configurable with the appropriate `col.` settings in the `group config` section.

You can lock a group with the `lockactivegroup` dispatcher in order to stop new window from entering this group. In addition, the `lockgroups` dispatcher can be used to toggle an independent global group lock that will prevent new window from entering any groups, regardless of their local group lock stat.

You can prevent a window from being added to group or becoming a group with the `denywindowfromgroup` dispatcher. `movewindoworgroup` will behave like `movewindow` if current active window or window in direction has this property set.

Workspaces

You have eight choices:

- ID: e.g. `1` , `2` , or `3`
- Relative ID: e.g. `+1` , `-3` or `+100`
- Relative workspace on monitor: e.g. `m+1` , `m-1` or `m+3`
- Relative workspace on monitor including empty workspaces: e.g. `r+1` or `r-3`
- Relative open workspace: e.g. `e+1` or `e-10`
- Name: e.g. `name:Web` , `name:Anime` or `name:Better anime`
- Previous workspace: `previous`
- First available empty workspace: `empty`
- Special Workspace: `special` or `special:name` for named special workspaces.

Warning

`special` is supported ONLY on `movetoworkspace` and `movetoworkspacesilent`. Any other dispatcher will result in undocumented behavior.

Important

Numerical workspaces (e.g. `1` , `2` , `13371337`) are allowed **ONLY** between 1 and

2147483647 (inclusive)

Neither 0 nor negative numbers are allowed.

Special Workspace

A special workspace is what is called a “scratchpad” in some other places. A workspace that you can toggle on/off on any monitor.

Note

You can define multiple named special workspaces, but the amount of those is limited to 97 at a time.

For example, to move a window/application to a special workspace you can use the following syntax:

```
bind = SUPER, C, movetoworkspace, special
#The above syntax will move the window to a special workspace upon
pressing 'SUPER'+'C'.
#To see the hidden window you can use the togglespecialworkspace
dispatcher mentioned above.
```

Executing with rules

The exec dispatcher supports adding rules. Please note some windows might work better, some worse. It records the PID of the spawned process and uses that. If your process e.g. forks and then the fork opens a window, this will not work.

The syntax is:

```
bind = mod, key, exec, [rules...] command
```

For example:

```
bind = SUPER, E, exec, [workspace 2 silent;float;noanim] kitty
```

The Hyprland Wiki, built with Hugo.



/Configuring/Dwindle Layout

Dwindle Layout

Dwindle is a BSPWM-like layout, where every window on a workspace is a member of a binary tree.

Quirks

Dwindle splits are NOT PERMANENT. The split is determined dynamically with the W/H ratio of the parent node. If W > H, it's side-by-side. If H > W, it's top-and-bottom. You can make them permanent by enabling `preserve_split`.

Config

category name: `dwindle`

| name | description | type | default |
|-----------------------------|--|------|---------|
| <code>pseudotile</code> | enable pseudotiling. Pseudotiled windows retain their floating size when tiled. [0/1/2] | bool | false |
| <code>force_split</code> | 0 -> split follows mouse, 1 -> always split to the left (new = left or top) 2 -> always split to the right (new = right or bottom) | int | 0 |
| <code>preserve_split</code> | if enabled, the split (side/top) will not change regardless of what happens to the container. | bool | false |

| name | description | type | default |
|------------------------------|--|-------------|----------------|
| smart_split | if enabled, allows a more precise control over the window split direction based on the cursor's position. The window is conceptually divided into four triangles, and cursor's triangle determines the split direction. This feature also turns on preserve_split. | bool | false |
| smart_resizing | if enabled, resizing direction will be determined by the mouse's position on the window (nearest to which corner). Else, it is based on the window's tiling position. | bool | true |
| permanent_direction_override | if enabled, makes the preselect direction persist until either this mode is turned off, another direction is specified, or a non-direction is specified (anything other than l,r,u/t,d/b) | bool | false |
| special_scale_factor | specifies the scale factor of windows on the special workspace [0 - 1] | float | 1 |
| split_width_multiplier | specifies the auto-split width multiplier | float | 1.0 |
| no_gaps_when_only | whether to apply gaps when there is only one window on a workspace, aka. smart gaps. (default: disabled - 0) no border - 1, with border - 2 [0/1/2] | int | 0 |

| name | description | type | default |
|-----------------------|--|-------------|----------------|
| use_active_for_splits | whether to prefer the active window or the mouse position for splits | bool | true |
| default_split_ratio | the default split ratio on window open. 1 means even 50/50 split. [0.1 - 1.9] | float | 1.0 |

Bind Dispatchers

| dispatcher | description | params |
|-------------------|--|---------------|
| pseudo | toggles the focused window's pseudo mode | none |

Layout messages

Dispatcher layoutmsg params:

| param | description | args |
|--------------|--|-------------|
| togglesplit | toggles the split (top/side) of the current window. <code>preserve_split</code> must be enabled for toggling to work. | none |
| preselect | A onetime override for the split direction. (valid for the next window to be opened, only works on tiled windows) | direction |

e.g.:

```
bind = SUPER, A, layoutmsg, preselect 1
```

The Hyprland Wiki, built with Hugo.



/Configuring/Environment Variables

Environment Variables

You can use the `env` keyword to set environment variables prior to the initialization of the Display Server, e.g.:

```
env = GTK_THEME, Nord
```

Important

Hyprland puts the raw string to the envvar with the `env` keyword. You should *not* add quotes around the values.

e.g.:

```
env = QT_QPA_PLATFORM, wayland
```

and **NOT**

```
env = QT_QPA_PLATFORM, "wayland"
```

Please avoid putting those environment variables in `/etc/environment`. That will cause all sessions (including Xorg ones) to pick up your wayland-specific environment on traditional Linux distros.

Hyprland Environment Variables

- `HYPRLAND_LOG_WLR=1` - Enables more verbose logging of wlroots.
- `HYPRLAND_NO_RT=1` - Disables realtime priority setting by Hyprland.
- `HYPRLAND_NO_SD_NOTIFY=1` - If systemd, disables the `sd_notify` calls.

Toolkit Backend Variables

- `GDK_BACKEND=wayland,x11` - GTK: Use wayland if available, fall back to x11 if not.
- `QT_QPA_PLATFORM="wayland;xcb"` - Qt: Use wayland if available, fall back to x11 if not.
- `SDL_VIDEODRIVER=wayland` - Run SDL2 applications on Wayland. Remove or set to `x11` if games that provide older versions of SDL cause compatibility issues
- `CLUTTER_BACKEND=wayland` - Clutter package already has wayland enabled, this variable will force Clutter applications to try and use the Wayland backend

XDG Specifications

- `XDG_CURRENT_DESKTOP=Hyprland`
- `XDG_SESSION_TYPE=wayland`
- `XDG_SESSION_DESKTOP=Hyprland`

XDG specific environment variables are often detected through portals and applications that may set those for you, however it is not a bad idea to set them explicitly.

Qt Variables

- `QT_AUTO_SCREEN_SCALE_FACTOR=1` - (From the Qt documentation) enables automatic scaling, based on the monitor's pixel density
- `QT_QPA_PLATFORM=wayland;xcb` - Tell Qt applications to use the Wayland backend, and fall back to x11 if Wayland is unavailable
- `QT_WAYLAND_DISABLE_WINDOWDECORATION=1` - Disables window decorations on Qt applications
- `QT_QPA_PLATFORMTHEME=qt5ct` - Tells Qt based applications to pick your theme from qt5ct, use with Kvantum.

NVIDIA Specific

To force GBM as a backend, set the following environment variables:

- `GBM_BACKEND=nvidia-drm`
- `__GLX_VENDOR_LIBRARY_NAME=nvidia`

See Archwiki Wayland Page for more details on those variables.

- `LIBVA_DRIVER_NAME=nvidia` - Hardware acceleration on NVIDIA GPUs

See Archwiki Hardware Acceleration Page for details and necessary values before setting this variable.

- `__GL_GSYNC_ALLOWED` - Controls if G-Sync capable monitors should use Variable Refresh Rate (VRR)

See Nvidia Documentation for details.

- `__GL_VRR_ALLOWED` - Controls if Adaptive Sync should be used. Recommended to set as "0" to avoid having problems on some games.
- `WLR_DRM_NO_ATOMIC=1` - use legacy DRM interface instead of atomic mode setting.
Might fix flickering issues.

Theming Related Variables

- `GTK_THEME` - Set a GTK theme manually, for those who want to avoid appearance tools such as lxappearance or nwg-look
- `XCURSOR_THEME` - Set your cursor theme. The theme needs to be installed and readable by your user.
- `XCURSOR_SIZE` - Set cursor size. See here for why you might want this variable set.



/Plugins/Event List

Event List

These are all the events that can be listened to using Event Hooks.

Complete list

Note

M: means std::unordered_map<std::string, std::any> following props are members.

| name | description | argument(s) |
|---------------|---|---|
| tick | fired on a tick, meaning once per (1000 / highestMonitorHz) ms | nullptr |
| activeWindow | fired on active window change | CWindow* |
| keyboardFocus | fired on keyboard focus change. Contains the newly focused surface | wlr_surface* |
| moveWorkspace | fired when a workspace changes its monitor | std::vector<void*>{CWorkspace*, CMonitor*} |
| focusedMon | fired on monitor focus change | CMonitor* |

| name | description | argument(s) |
|------------------|--|---|
| moveWindow | fired when a window changes workspace | std::vector<void*>{CWindow*, CWorkspace*} |
| openLayer | fired when a LS is mapped | CLayerSurface* |
| closeLayer | fired when a LS is unmapped | CLayerSurface* |
| openWindow | fired when a window is mapped | CWindow* |
| closeWindow | fired when a window is unmapped | CWindow* |
| urgent | fired when a window requests urgent | CWindow* |
| minimize | fired when a window requests a minimize status change. Second param is 1 or 0 | std::vector<void*>{CWindow*, uint64_t} |
| monitorAdded | fired when a monitor is plugged in | CMonitor* |
| monitorRemoved | fired when a monitor is unplugged | CMonitor* |
| createWorkspace | fired when a workspace is created | CWorkspace* |
| destroyWorkspace | fired when a workspace is destroyed | CWorkspace* |
| fullscreen | fired when a window changes fullscreen | CWindow* |

| name | description | argument(s) |
|--------------------|--|--|
| | state | |
| changeFloatingMode | fired when a window changes float state | CWindow* |
| workspace | fired on a workspace change (only ones explicitly requested by a user) | CWorkspace* |
| submap | fired on a submap change | std::string |
| mouseMove | fired when the cursor moves. Param is coords. | const Vector2D |
| mouseButton | fired on a mouse button press | wlr_pointer_button_event* |
| mouseAxis | fired on a mouse axis event | M: event : wlr_pointer_axis_event* |
| touchDown | fired on a touch down event | wlr_touch_down_event* |
| touchUp | fired on a touch up event | wlr_touch_up_event* |
| touchMove | fired on a touch motion event | wlr_touch_motion_event* |
| activeLayout | fired on a keyboard layout change. String pointer temporary, not guaranteed after execution of the handler finishes. | std::vector<void*>{SKeyboard*, std::string*} |

| name | description | argument(s) |
|-----------------|--|--|
| preRender | fired before a frame for a monitor is about to be rendered | CMonitor* |
| screencast | fired when the screencopy state of a client changes. Keep in mind there might be multiple separate clients. | std::vector<uint64_t>{state, framesInHalfSecond, owner} |
| render | fired at various stages of rendering to allow your plugin to render stuff. See src/SharedDefs.hpp for a list with explanations | eRenderStage |
| windowtitle | emitted when a window title changes. | CWindow* |
| configReloaded | emitted after the config is reloaded | nullptr |
| preConfigReload | emitted before a config reload | nullptr |
| keyPress | emitted on a key press | M: event : wlr_keyboard_key_event* , keyboard : SKeyboard* |

The Hyprland Wiki, built with Hugo.



/Configuring/Expanding Functionality

Expanding Functionality

Hyprland exposes two powerful sockets for you to use.

The first, socket1, can be fully controlled with `hypctl`, see its usage here.

The second, socket2, sends events for certain changes / actions and can be used to react to different events. See its description here.

Example script

This bash script will change the outer gaps to 20 if the currently focused monitor is DP-1, and 30 otherwise.

```
#!/bin/bash

function handle {
    if [[ ${1:0:10} == "focusedmon" ]]; then
        if [[ ${1:12:4} == "DP-1" ]]; then
            hypctl keyword general:gaps_out 20
        else
            hypctl keyword general:gaps_out 30
        fi
    fi
}

socat - "UNIX-CONNECT:/tmp/hypr/$HYPRLAND_INSTANCE_SIGNATURE/.socket2.sock" |
while read -r line; do handle "$line"; done
```

[/Faq](#)

Faq

Table of Contents

- Table of Contents
- My apps are pixelated
- Nothing renders / screen is empty / crash on opening first app
- Me cursor no render?
- My external monitor is blank / doesn't render / receives no signal (laptop)
- How do I screenshot?
- Screenshare / OBS no worky
- How do I change my wallpaper?
- How heavy is this?
- My monitor no worky
- My monitor has flickering brightness when I turn on VRR
- How do I update?
- How do I screen lock?
- How do I change me mouse cursor?
- GTK Settings no work / whatever
- My [program name] is freezing
- Waybar workspaces no worky???
- How do I autostart my favorite apps?
- How do I move my favorite workspaces to a new monitor when I plug it in?
- My tablet no worky??
- Some of my apps take a really long time to open...?
- How do I export envvars for Hyprland?
- How to disable middle-click paste?
- How do I make Hyprland draw as little power as possible on my laptop?
- How to fix games with window dancing?

- My apps take a long time to start / can't screenshare
- My screenshot utilities won't work with multiple screens
- I cannot bind SUPER as my mod key on my laptop
- My VM doesn't receive keybinds I have set in Hyprland
- Some of my drop-down/pop-up windows in apps disappear

My apps are pixelated

This just means they are running through XWayland, which physically cannot scale by fractional amounts.

To force them to run in wayland-native mode, see the Master Tutorial.

If they can't, see the XWayland page.

Nothing renders / screen is empty / crash on opening first app

Possible causes:

Your themes are not set up properly, making apps crash.

Use something like `qt6ct` (Qt) and `nwg-look` (GTK) (*for GTK you can also set up themes with envvars) to set up your themes.

Your PC is very, very old.

In that case, see the Installation Page and try compiling with `LEGACY_RENDERER`

For more info about bugs and crashes, see this wiki page

Me cursor no render?

Are you on NVIDIA? If so, then you have been a naughty boy and haven't listened to my tips on other pages. Use the `WLR_NO_HARDWARE_CURSORS=1` environment variable.

My external monitor is blank / doesn't render / receives no signal (laptop)

For Nvidia graphics - This issue appears to be resolved when using Nvidia Drivers 525.60.11 or later, but it may persist with older drivers.

Outside those, there is a way to fix it that *might* work for you though:

Option 1: Use *only* the external monitor

By using `WLR_DRM_DEVICES=/dev/dri/card1` (or `card0`) you can force Hyprland to use only your dGPU, meaning your laptop's screen will be gone but your external one will work.

Option 2: Use all outputs, at the cost of battery life.

By switching your laptop to only use the dGPU in the BIOS, you *might* be able to get everything to work, at the cost of high battery usage.

Please note these are highly model-specific and might or might not work. If they don't, you're unfortunately out of luck.

You might try a USB-C to HDMI adapter though, maybe that could route the external monitor through the iGPU.

How do I screenshot?

Install `grim` and `slurp`

Use a keybind (or execute) `grim -g "$(slurp)"`, select a region. A screenshot will pop into your `~/Pictures/` (You can configure `grim` and `slurp`, see their GitHub pages).

If you want those screenshots to go directly to your clipboard, consider using `wl-copy`, from `wl-clipboard`. Here's an example binding: `bind = , Print, exec, grim -g "$(slurp -d)" - | wl-copy` For a more complete utility, try our own screenshotting utility: Grimblast.

For recording videos, `wf-recorder`, `wl-screenrec` or `OBS Studio` could be used.

Screenshare / OBS no worky

Check Screensharing.

Also install `qt6-wayland` if you plan to use obs.

How do I change my wallpaper?

See Wallpapers.

How heavy is this?

Not that much heavier than Xorg. If you want maximum performance, consider turning off the blur and animations.

My monitor no worky

Try changing the mode in your config. If your preferred one doesn't work, try a lower one. A good way to list all modes is to get `wlr-randr` and do a `wlr-randr --dryrun`

My monitor has flickering brightness when I turn on VRR

Change the VRR option to `2` (fullscreen), so that it is only used in games. This happens because the brightness on some monitors can depends on the refresh rate, and rapidly changing refresh rates (for example, when the screen momentarily updates after pressing a key) will cause rapid changes in brightness.

How do I update?

Open a terminal where you cloned the repo.

```
git pull  
make all && sudo make install
```

If you are using the AUR (`hyprland-git`) package, you will need to cleanbuild to update the package. Paru has been problematic with updating before, use Yay.

How do I screen lock?

Use a wayland-compatible locking utility using WLR protocols, e.g. `swaylock`.

How do I change my mouse cursor?

1. Set the GTK cursor using `nwg-look`.
2. Add `exec-once=hyprctl setcursor [THEME] [SIZE]` to your config and restart Hyprland.

If using flatpak, run `flatpak override --env=~/themes:ro --env=~/icons:ro --user` and put your themes in both `/usr/share/themes` and `~/.themes`, and put your icons and cursors in both `/usr/share/icons` and `~/.icons`.

For Qt applications, Hyprland exports `XCURSOR_SIZE` as 24, which is the default. You can overwrite this by exporting `XCURSOR_SIZE` to a different value with `env`.

You can also try running `gsettings set org.gnome.desktop.interface cursor-theme 'theme-name'` or adding it after `exec-once=` in your config.

If you do not want to install a GTK settings editor, change the config files according to the XDG specification (Arch Wiki link). Make sure to also edit `~/.config/gtk-4.0/settings.ini` and `~/.gtkrc-2.0` if *not* using a tool (like `nwg-look`).

GTK Settings no work / whatever

<https://github.com/swaywm/sway/wiki/GTK-3-settings-on-Wayland>

My [program name] is freezing

Make sure you have a notification daemon running, for example `dunst`. Autostart it with the `exec-once` keyword.

Waybar workspaces no worky???

Waybar has a set of caveats or settings that you need to be aware of. See Status bars for

solutions.

How do I autostart my favorite apps?

Using the window rules to assign apps to workspace you can open a bunch of applications on various workspaces. The following method will start these apps silently (i.e. without the flickering from workspace to workspace).

Put the following in your `hyprland.conf` : (example)

```
exec-once=[workspace 1 silent] kitty
exec-once=[workspace 1 silent] subl
exec-once=[workspace 3 silent] mailspring
exec-once=[workspace 4 silent] firefox
```

How do I move my favorite workspaces to a new monitor when I plug it in?

if you want workspaces to automatically go to a monitor upon connection, use the following:

In `hyprland.conf`:

```
exec-once=handle_monitor_connect.sh
```

where `handle_monitor_connect.sh` is: (example)

```
#!/bin/sh

handle() {
    case $1 in
        monitoradded*)
            hyprctl dispatch moveworkspacetomonitor "1 1"
            hyprctl dispatch moveworkspacetomonitor "2 1"
            hyprctl dispatch moveworkspacetomonitor "4 1"
            hyprctl dispatch moveworkspacetomonitor "5 1"
    esac
}

socat - "UNIX-CONNECT:/tmp/hypr/${HYPRLAND_INSTANCE_SIGNATURE}/.socket2.sock"
| while read -r line; do handle "$line"; done
```

if you want workspaces 1 2 4 5 to go to monitor 1 when connecting it.

Please note this requires `socat` to be installed.

My tablet no worky??

Use Open Tablet Driver to configure your tablet. In the future it will be supported in the config. Until then, OTD is the way to go.

Some of my apps take a really long time to open...?

`~/.config/hypr/hyprland.conf`

```
exec-once=dbus-update-activation-environment --systemd WAYLAND_DISPLAY  
XDG_CURRENT_DESKTOP
```

Make sure that your portals launch *after* this gets executed. For some people, they might launch before that has happened.

In such cases, a script like this:

```
#!/bin/bash  
sleep 4  
killall -e xdg-desktop-portal-wlr  
killall xdg-desktop-portal  
/usr/lib/xdg-desktop-portal-wlr &  
sleep 4  
/usr/lib/xdg-desktop-portal &
```

launched with `exec-once` should fix all issues. Adjust the sleep durations to taste.

How do I export envvars for Hyprland?

See Environment Variables

The `env` keyword is used for this purpose. For example:

```
env = XDG_CURRENT_DESKTOP,Hyprland
```

How to disable middle-click paste?

You can simply intercept the middle-click action all together, via hyprland binds for example. The drawbacks to this solution are that 1. it disables the rest of the functionality of the middle-click action, such as auto scroll, closing browser tabs, etc., and 2. many applications (such as kitty) manually intercept the middle-click events and bind them to paste from the primary buffer themselves, bypassing the solution altogether. For this solution, add this bind to your config:

```
bind = , mouse:274, exec, ; . Note that the exact bindcode may vary, so you may want to check it with wev first.
```

- ▶ Alternative method using wl-paste (warning: major power consumption)

How do I make Hyprland draw as little power as possible on my laptop?

Useful Optimizations:

- decoration:blur = false and decoration:drop_shadow = false to disable fancy but battery hungry effects.
- misc:vfr = true , since it'll lower the amount of sent frames when nothing is happening on-screen.

How to fix games with window dancing?

Read this trick.

My apps take a long time to start / can't screenshare

See The XDPH Page.

You most likely have multiple portal impls / an impl is failing to launch.

My screenshot utilities won't work with multiple screens

Some programs like flameshot (currently) has limited wayland support so on most Wayland compositors, you will have to do few tweaks. For Hyprland, you can add these window rules to your config to make said programs work with both of your screens.

```
windowrulev2=move 0 0,title:^(flameshot)  
windowrulev2=nofullscreenrequest,title:^(flameshot)
```

I cannot bind SUPER as my mod key on my laptop

Many laptops have a built-in function to toggle `SUPER` between single key press mode and hold mode. This is usually indicated by a padlock on the `SUPER` key.

First, install and run `wev`, then press `SUPER`. If you see a key press event followed by an instant key release event, then it's likely your `SUPER` key is set to single press mode.

On most laptops this can be fixed by pressing `FN+SUPER` and verified in `wev`. You should be able to hold `SUPER` and not see an instant release event. In case `FN+SUPER` doesn't work, consult your laptop's manual.

My VM doesn't receive keybinds I have set in Hyprland

This is expected, as Hyprland takes precedence.

A simple fix is to create an empty “passthrough” submap:

```
bind = MOD,KEY,submap,passthru  
submap = passthru  
bind = SUPER,Escape,submap,reset  
submap = reset
```

set `MOD` and `KEY` to desired values.

By pressing the selected combo you will enter a mode where hyprland ignores your keybinds and passes them on to the vm.

Then, pressing `SUPER + Escape` will leave that mode.

Some of my drop-down/pop-up windows in apps disappear

In some apps like Steam or VSCode, the drop-down windows may disappear if you hover over them. This can be fixed with window rules.

First, find the title and class of the pop-up window with `hyprctl clients`. You can try something like `sleep 3 && hyprctl clients` so you have time to open the pop-up. It should look something like this:

```
Window 55d794495400 -> :  
...  
class: [CLASS here]  
title: [TITLE here]  
...
```

If the pop-up disappears as you hover over it, you can add to your config:

```
windowrulev2 = stayfocused, title:^(TITLE)$, class:^(CLASS)$
```

This has a downside of not being able to click on anything in the main UI until you've interacted with the pop-up.

If the pop-up disappears immediately, you can use:

```
windowrulev2 = minsize 1 1, title:^(TITLE)$, class:^(CLASS)$
```



 /Plugins/Getting Started

Getting Started

This page documents the basics of making your own Hyprland plugin from scratch.

- How do plugins work?
- Prerequisites
- Making your first plugin
 - The basic parts of the plugin
 - Setting up a development environment
 - More advanced stuff

How do plugins work?

Plugins are basically dynamic objects loaded by Hyprland. They have (almost) full access to every part of Hyprland's internal process, and as such, can modify and change way more than a script.

Prerequisites

In order to write a Hyprland plugin, you will need:

- Knowledge of C++
- The ability to read
- A rough understanding of the Hyprland internals (you *can* learn this alongside your development work)

Making your first plugin

Open your favorite code editor.

Make a new directory, in this example we will use `MyPlugin`.

→ **If you have the Hyprland headers**

If you install with `make install`, you should have the headers. In that case, no further action is required.

→ **If you don't have the Hyprland source cloned**

Clone the Hyprland source code to a subdirectory, in our example `MyPlugin/Hyprland`. Run `cd Hyprland && make all && sudo make installheaders && cd . . .`

Now that you have the Hyprland sources set up, you can either start from scratch if you know how, or take a look at some simple plugins in the official plugins repo like for example `csgo-vulkan-fix` or `hyprwinwrap`.

The basic parts of the plugin

Starting from the top, you will have to include the plugin API:

```
#include <hyprland/src/plugins/PluginAPI.hpp>
```

Feel free to take a look at the header. It contains a bunch of useful comments.

We also create a global pointer for our handle:

```
inline HANDLE PHANDLE = nullptr;
```

we will initialize it in our plugin init function later. It serves as an internal “ID” of our plugin.

Then, there is the API version method:

```
// Do NOT change this function.
APICALL EXPORT std::string PLUGIN_API_VERSION() {
    return HYPRLAND_API_VERSION;
}
```

This method will tell Hyprland what API version was used to compile this plugin. Do NOT change it. It will be set automatically when compiling to the correct value.

Skipping over some example handlers, we have two important functions:

```
APICALL EXPORT PLUGIN_DESCRIPTION_INFO PLUGIN_INIT(HANDLE handle) {
    PHANDLE = handle;
```

```
// ...  
  
    return {"MyPlugin", "An amazing plugin that is going to change the  
world!", "Me", "1.0"};  
}  
  
APICALL EXPORT void PLUGIN_EXIT() {  
    // ...  
}
```

The first method will be called when your plugin gets initialized (loaded)

You can, and probably should, initialize everything you may want to use in there.

It's worth noting that adding config variables is *only* allowed in this function.

The plugin init function is *required*.

The return value should be the `PLUGIN_DESCRIPTION_INFO` struct which lets Hyprland know about your plugin's name, description, author and version.

Make sure to store your `HANDLE` as it's going to be required for API calls.

The second method is not required, and will be called when your plugin is being unloaded by the user.

If your plugin is being unloaded because it committed a fault, this function will *not* be called.

You do not have to unload layouts, remove config options, remove dispatchers, window decorations or unregister hooks in the exit method. Hyprland will do that for you.

Setting up a development environment

In order to make your life easier, it's a good idea to work on a nested debug Hyprland session.

Enter your Hyprland directory and run `make debug`

Make a copy of your config in `~/.config/hypr` called `hyprlandd.conf`.

Remove *all* `exec=` or `exec-once=` directives from your config.

recommended: Change the modifier for your keybinds (e.g. `SUPER -> ALT`)

Add this line:

```
monitor = WL-1, 1920x1080, 0x0, 1
```

Launch the output `Hyprland` binary in `./build/` when logged into a `Hyprland` session.

A new window should open with `Hyprland` running inside of it. You can now run your plugin in the nested session without worrying about nuking your actual session, and also being able to debug it easily.

See more info in the Contributing Section

More advanced stuff

Take a look at the `src/plugins/PluginAPI.hpp` header. It has comments to every method to let you know what it is.

For more explanation on a few concepts, see Advanced and Plugin Guidelines

The Hyprland Wiki, built with Hugo.



[/Useful Utilities/Hyprrland Desktop Portal](#)

Hyprrland Desktop Portal

An XDG Desktop Portal (later called XDP) is a program that lets other applications communicate swiftly with the compositor through D-Bus.

It's used for stuff like e.g. opening file pickers, screen sharing.

On Wayland, it also requires an implementation. For Hyprrland, you'd usually use `xdg-desktop-portal-wlr` (later called XDPW)

Unfortunately, due to various reasons the `-wlr` portal is inferior to the KDE or Gnome ones.

In order to bridge the gap, Hyprrland has its own fork of XDPW that has more features, called `xdg-desktop-portal-hyprrland`. (later called XDPH)

Important

You don't **need** XDPH. Hyprrland will work with XDPW, but XDPH has more features, like e.g. window sharing.

XDPH will work on other wlroots-based compositors, although limited to the XDPW features (other will be disabled)

Installing

[Arch Linux](#) [Gentoo](#) [Manual](#)

```
pacman -S xdg-desktop-portal-hyprrland
```

or, for -git:

```
yay -S xdg-desktop-portal-hyprrland-git
```

Tip

XDPH doesn't implement a file picker. For that, I recommend installing `xdg-desktop-portal-gtk` alongside XDPH.

Usage

Should start automatically.

The most basic way of telling everything is OK is by trying to screenshare anything, or open OBS and select pipewire source. If XDPH is running, a qt menu will pop up asking you what to share.

If it doesn't, and you get e.g. slurp, then XDPW is launching. In that case, try removing XDPW.

XDPH will work on other wlroots compositors, but features available only on Hyprland will not work (e.g. window sharing)

For a nuclear option, you can use this script and `exec-once` it:

```
#!/bin/bash
sleep 1
killall -e xdg-desktop-portal-hyprland
killall -e xdg-desktop-portal-wlr
killall xdg-desktop-portal
/usr/lib/xdg-desktop-portal-hyprland &
sleep 2
/usr/lib/xdg-desktop-portal &
```

adjust the paths if incorrect.

Share picker doesn't use the system theme

Try one or both:

```
dbus-update-activation-environment --systemd --all
systemctl --user import-environment QT_QPA_PLATFORMTHEME
```

If it works, add it to your config in an `exec-once`.

Debugging

If you get long app launch times, or screensharing does not work, consult the logs.

```
systemctl --user status xdg-desktop-portal-hyprland
```

if you see a crash, it's most likely you are missing `qt6-wayland` and/or `qt5-wayland`.

The Hyprland Wiki, built with Hugo.



 /Nix/Hyprland on Home Manager

Hyprland on Home Manager

For a list of available options, check the Home Manager options.

Note

- (*Required*) *NixOS Module*: enables critical components needed to run Hyprland properly
- (*Optional*) *Home-manager module*: lets you declaratively configure Hyprland

Installation

Home Manager Flakes No flakes (with flake-compat)

Home Manager has options for Hyprland without needing to import the Flake module.

```
{  
  wayland.windowManager.hyprland.enable = true;  
}
```

};

Usage

Once the module is enabled, you can use it to declaratively configure Hyprland. Here is an example config, made to work with either the upstream Home Manager module, or the flake-based Home Manager module.

```
# home.nix  
{  
  wayland.windowManager.hyprland.settings = {  
    "$mod" = "SUPER";
```

```

bind =
[
    "$mod, F, exec, firefox"
    ", Print, exec, grimblast copy area"
]
++ (
    # workspaces
    # binds $mod + [shift+] {1..10} to [move to] workspace {1..10}
    builtins.concatLists (builtins.genList (
        x: let
            ws = let
                c = (x + 1) / 10;
            in
                builtins.toString (x + 1 - (c * 10));
            in [
                "$mod, ${ws}, workspace, ${toString (x + 1)}"
                "$mod SHIFT, ${ws}, movetoworkspace, ${toString (x + 1)}"
            ]
        )
    )
    10)
);
}
}

```

Plugins

Hyprland plugins can be added through the `plugins` option:

```

wayland.windowManager.hyprland.plugins = [
    inputs.hyprland-plugins.packages.${pkgs.system}.hyprbars
    "/absolute/path/to/plugin.so"
];

```

For examples on how to build Hyprland plugins using nix see the official [plugins](#).

Fixing problems with themes

If your themes for mouse cursor, icons or windows don't load correctly, try setting them with `home.pointerCursor` and `gtk.theme`, which enable a bunch of compatibility options that should make the themes load in all situations.

Example configuration:

```

home.pointerCursor = {

```

```
gtk.enable = true;
# x11.enable = true;
package = pkgs.bibata-cursors;
name = "Bibata-Modern-Classic";
size = 16;
};

gtk = {
  enable = true;
  theme = {
    package = pkgs.flat-remix-gtk;
    name = "Flat-Remix-GTK-Grey-Darkest";
  };
};

iconTheme = {
  package = pkgs.gnome.adwaita-icon-theme;
  name = "Adwaita";
};

font = {
  name = "Sans";
  size = 11;
};
};
```

The Hyprland Wiki, built with Hugo.



 /Nix/Hyprland on Nix Os

Hyprland on Nix Os

The NixOS module enables critical components needed to run Hyprland properly, such as: polkit, xdg-desktop-portal-hyprland, graphics drivers, fonts, dconf, xwayland, and adding a proper Desktop Entry to your Display Manager.

Make sure to check out the options of the NixOS module.

Note

- *(Required) NixOS Module*: enables critical components needed to run Hyprland properly
- *(Optional) Home-manager module*: lets you declaratively configure Hyprland

| Nixpkgs | Flake package | Flake package, Nix stable |
|---------|---------------|---------------------------|
|---------|---------------|---------------------------|

```
# configuration.nix

{pkgs, ...}: {
    programs.hyprland.enable = true;
}
```

This will use the Hyprland version that Nixpkgs has.

Fixing problems with themes

If your themes for mouse cursor, icons or windows don't load correctly, see the relevant section in Hyprland on Home Manager.

The Hyprrland Wiki, built with Hugo.

Hyprland Wiki

 /Getting Started/Installation

Installation

Foreword

Due to their proprietary nature, Nvidia GPUs have limited compatibility with Hyprland. If you want to try Hyprland on Nvidia regardless (many people have reported successes), follow the Nvidia page after installing Hyprland.

Distros

Arch, NixOS and openSUSE Tumbleweed are very supported. For any other distro (not based on Arch/Nix) you might have varying amounts of success. However, since Hyprland is extremely bleeding-edge, distros like Pop!_OS, Ubuntu, etc. might have **major** issues running Hyprland.

Installation

Installing Hyprland is very easy. Either you install it from your local package provider (if they provide pkgs for Hyprland) or you install/build it yourself.

note

This project is under development and is constantly changing. If you want to keep up to date with the latest commits, please consider updating your packages with `yay -Syu --devel`, or your other preferred package manager.

Packages

WARNING: I do not maintain any packages. If they are broken, try building from source first.

| | | | | | |
|---------------|-------------|------------|---------|---------|----------|
| Arch Linux | Nix | openSUSE* | Fedora* | Gentoo* | FreeBSD* |
| Ubuntu 23.04* | Void Linux* | Slackware* | | | |

hyprland-git (AUR) - compiles from latest source
hyprland - binary x86 tagged release

*** Unofficial, no official support is provided. These instructions are community-driven, and no guarantee is provided for their validity.**

Manual (Releases, Linux-only)

Download the most recent release.

copy the binary (Hyprland) to `/usr/bin/ .`

copy hyprctl to `/usr/bin/ .`

copy hyprpm to `/usr/bin/ .`

copy the wlroots .so (`libwlroots.so.XX032`) to `/usr/lib/ .`

copy the desktop entry (`example/hyprland.desktop`) to `/usr/share/wayland-sessions/`

the example config is in `example/hyprland.conf` .

For updating later on, you can overwrite the binaries (hyprctl, hyprland and libwlroots), you don't need to update anything else.

Manual (Manual Build)

Arch dependencies:

```
yay -S gdb ninja gcc cmake meson libxcb xcb-proto xcb-util xcb-util-keysyms  
libxfixes libx11 libcomposite xorg-xinput libxrender pixman wayland-protocols  
cairo pango seatd libxkbcommon xcb-util-wm xorg-xwayland libinput libliftoff  
libdisplay-info cpio tomlplusplus
```

(Please make a pull request or open an issue if any packages are missing from the list)

openSUSE dependencies:

```
zypper in gcc-c++ git meson cmake "pkgconfig(cairo)" "pkgconfig(egl)"  
"pkgconfig(gbm)" "pkgconfig(gl)" "pkgconfig(glesv2)" "pkgconfig(libdrm)"  
"pkgconfig(libinput)" "pkgconfig(libseat)" "pkgconfig(libudev)"  
"pkgconfig(pango)" "pkgconfig(pangocairo)" "pkgconfig(pixman-1)"  
"pkgconfig(vulkan)" "pkgconfig(wayland-client)" "pkgconfig(wayland-protocols)"  
"pkgconfig(wayland-scanner)" "pkgconfig(wayland-server)" "pkgconfig(xcb)"  
"pkgconfig(xcb-icccm)" "pkgconfig(xcb-renderutil)" "pkgconfig(xkbcommon)"  
"pkgconfig(xwayland)" "pkgconfig(xcb-errors)" glslang-devel Mesa-libGLESv3-  
devel tomplusplus-devel
```

(this should also work on RHEL/Fedora if you remove Mesa-libGLESv3-devel and pkgconfig(xcb-errors))

FreeBSD dependencies:

```
pkg install git pkgconf gmake gcc evdev-proto cmake wayland-protocols wayland  
libglvnd libxkbcommon libinput cairo pango pixman libxcb  
pkg install meson jq `pkg rquery %dn wlroots` hwdetect libdisplay-info  
libliftoff  
export CC=gcc CXX=g++ LDFLAGS="-static-libstdc++ -static-libgcc"
```

Ubuntu 23.04 dependencies: refer to the Ubuntu tab above

Please note Hyprland builds wlroots . Make sure you have the dependencies of wlroots installed, you can make sure you have them by installing wlroots separately (Hyprland doesn't mind)

Also note that Hyprland uses the C++23 standard, so both your compiler and your C++ library has to support that (gcc>=13.0.0 or clang>=15). On Clang-based systems libc++ may be used by default, so until libc++ supports C++23 you have to pass -stdlib=libstdc++ or switch to GCC.

CMake (recommended)

```
git clone --recursive https://github.com/hyprwm/Hyprland  
cd Hyprland  
make all && sudo make install
```

CMake is always recommended as it's the intended way Hyprland should be installed.

Meson

```
meson subprojects update --reset  
meson setup build  
ninja -C build  
ninja -C build install --tags runtime,man
```

Refer to Debugging to see how to build & debug.

Crash on launch

See Crashes and Bugs.

Custom installation (legacy renderer, etc)

cd into the hyprland repo.

for legacy renderer:

```
make legacyrenderer && sudo cp ./build/Hyprland /usr/bin && sudo cp ./example  
/hyprland.desktop /usr/share/wayland-sessions
```

please note the legacy renderer may not support some graphical features.

Any other config: (replace [PRESET] with your preset, release debug legacyrenderer
legacyrendererdebug)

```
make [PRESET] && sudo cp ./build/Hyprland /usr/bin && sudo cp ./example  
/hyprland.desktop /usr/share/wayland-sessions
```

Custom Build flags

To apply custom build flags, you'll have to ditch make.

Supported custom build flags:

```
NO_XWAYLAND - Removes XWayland support  
NO_SYSTEMD - Removes systemd dependencies
```

How to?

Go to the root repo.

Then, configure CMake:

```
mkdir -p build && cmake --no-warn-unused-cli -DCMAKE_BUILD_TYPE:STRING=Release  
-D<YOUR_FLAG>:STRING=true -H./ -B./build -G Ninja
```

Change `<YOUR_FLAG>` to one of the custom build flags. You **are allowed to** use multiple at once, then add another `-D<YOUR_FLAG_2>:STRING=true`

You can of course also change the `BUILD_TYPE` to `Debug`.

Now, build:

```
cmake --build ./build --config Release --target all -j $(nproc)
```

If you configured in `Debug`, change the `--config` to `Debug` as well.

Now, of course, install manually.

```
sudo cp ./build/Hyprland /usr/bin && sudo cp ./example/hyprland.desktop  
/usr/share/wayland-sessions
```

Lastly, copy `hyprctl`, `hyprpm`, and `wlroots` as mentioned here

The Hyprland Wiki, built with Hugo.



/IP C

IP C

Hyprland exposes 2 UNIX Sockets, for controlling / getting info about Hyprland via code / bash utilities.

Hyprland Instance Signature (HIS)

```
echo $HYPRLAND_INSTANCE_SIGNATURE
```

/tmp/hypr/[HIS]/.socket.sock

Used for hyprctl-like requests. See the Hyprctl page for commands.

basically, write [flag(s)]/command args .

/tmp/hypr/[HIS]/.socket2.sock

Used for events. Hyprland will write to each connected client live events like this:

EVENT>>DATA\n (\n is a linebreak)

e.g.: workspace>>2

Events list

| name | description | data |
|-----------|----------------------|---------------|
| workspace | emitted on workspace | WORKSPACENAME |

| name | description | data |
|----------------|---|--|
| | change. Is emitted ONLY when a user requests a workspace change, and is not emitted on mouse movements (see activemon) | |
| focusedmon | emitted on the active monitor being changed. | MONNAME, WORKSPACENAME |
| activewindow | emitted on the active window being changed. | WINDOWCLASS, WINDOWTITLE |
| activewindowv2 | emitted on the active window being changed. | WINDOWADDRESS |
| fullscreen | emitted when a fullscreen status of a window changes. | 0/1 (exit fullscreen / enter fullscreen) |
| monitorremoved | emitted when a monitor is removed (disconnected) | MONITORNAME |
| monitoradded | emitted when a monitor is added (connected) | MONITORNAME |

| name | description | data |
|------------------|---|---|
| createworkspace | emitted when a workspace is created | WORKSPACENAME |
| destroyworkspace | emitted when a workspace is destroyed | WORKSPACENAME |
| moveworkspace | emitted when a workspace is moved to a different monitor | WORKSPACENAME, MONNAME |
| renameworkspace | emitted when a workspace is renamed | WORKSPACEID, NEWNAME |
| activespecial | emitted when the special workspace opened in a monitor changes (closing results in an empty WORKSPACENAME) | WORKSPACENAME, MONNAME |
| activelayout | emitted on a layout change of the active keyboard | KEYBOARDNAME, LAYOUTNAME |
| openwindow | emitted when a window is opened | WINDOWADDRESS , WORKSPACENAME , WINDOW(|
| closewindow | emitted when a window is closed | WINDOWADDRESS |
| movewindow | emitted when a window is moved | WINDOWADDRESS , WORKSPACENAME |

| name | description | data |
|--------------------|--|--------------------------|
| | to a workspace | |
| openlayer | emitted when a layerSurface is mapped | NAMESPACE |
| closelayer | emitted when a layerSurface is unmapped | NAMESPACE |
| submap | emitted when a keybind submap changes. Empty means default. | SUBMAPNAME |
| changefloatingmode | emitted when a window changes its floating mode. FLOATING is either 0 or 1. | WINDOWADDRESS , FLOATING |
| urgent | emitted when a window requests an urgent state | WINDOWADDRESS |
| minimize | emitted when a window requests a change to its minimized state. MINIMIZED is either 0 or 1. | WINDOWADDRESS, MINIMIZED |
| screencast | emitted when a screencopy state of a client changes. Keep in mind there might be multiple | STATE, OWNER |

| name | description | data |
|-----------------|--|---------------|
| | separate clients. State is 0/1, owner is 0 - monitor share, 1 - window share | |
| windowtitle | emitted when a window title changes. | WINDOWADDRESS |
| ignoregrouplock | emitted when ignoregrouplock is toggled. | 0/1 |
| lockgroups | emitted when lockgroups is toggled. | 0/1 |
| configreloaded | emitted when the config is done reloading | empty |

Warning

A fullscreen event is not guaranteed to fire on/off once in succession. A window might do for example 3 requests to be fullscreen'd, which would result in 3 fullscreen events.

How to use socket2 with bash

example script using socket2 events with bash and `socat` :

```
#!/bin/sh

handle() {
  case $1 in
    monitoradded*) do_something ;;
    focusedmon*) do_something_else ;;
  esac
}
```

```
}
```

```
socat -U - UNIX-CONNECT:/tmp/hypr/$HYPRLAND_INSTANCE_SIGNATURE/.socket2.sock |  
while read -r line; do handle "$line"; done
```

The Hyprland Wiki, built with Hugo.



/Configuring/Keywords

Keywords

Keywords are not variables, but “commands” for more advanced configuring. On this page, you will be presented with some that do not deserve their own page.

See the sidebar for more keywords to control binds, animations, monitors, et cetera.

Important

Please remember, that for ALL arguments separated by a comma, if you want to leave one of them empty, you cannot reduce the number of commas, *unless told otherwise in a specific section:*

```
three_param_keyword = A, B, C # OK  
three_param_keyword = A, C # NOT OK  
three_param_keyword = A, , C # OK  
three_param_keyword = A, B, # OK
```

Table of contents

- Table of contents
- Executing
- Defining variables
- Sourcing (multi-file)
- Gestures
- Per-device input configs
- Wallpapers
- Blurring layerSurfaces
- Setting the environment

Executing

you can execute a shell script on startup of the compositor or on each time it's reloaded.

`exec-once=command` will execute only on launch

`exec=command` will execute on each reload

Defining variables

You can define your own custom variables like this:

```
$VAR = value
```

for example:

```
$MyFavoriteGame = Among Us
```

then, to use them, simply use them. For example:

```
col.active_border=$MyColor
```

You ARE allowed to do this:

```
col.active_border=ff$MyRedValue1111
```

Sourcing (multi-file)

Use the `source` keyword to source another file.

For example, in your `hyprland.conf` you can:

```
source=~/config/hypr/myColors.conf
```

And Hyprland will enter that file and parse it like a Hyprland config.

Please note it's LINEAR. Meaning lines above the `source=` will be parsed first, then lines

inside `~/.config/hypr/myColors.conf`, then lines below.

Gestures

Use something like `libinput-gestures`, with `hyprctl` if you want to expand Hyprland's gestures beyond what's offered in Variables.

Per-device input configs

Per-device config options will overwrite your options set in the `input` section. It's worth noting that ONLY values explicitly changed will be overwritten.

In order to apply per-device config options, make a new category like this:

```
device:name {  
}  
}
```

The `name` can be easily obtained by doing `hyprctl devices`.

Inside of it, put your config options. All options from the `input` category (and all subcategories, e.g. `input:touchpad`) can be put inside, **EXCEPT**:

`force_no_accel`, `follow_mouse`, `float_switch_override_focus`, `scroll_factor`

Properties that change names:

```
touchdevice:transform -> transform  
touchdevice:output -> output
```

You can also use the `output` setting for tablets to bind them to outputs. Remember to use the name of the `Tablet` and not `Tablet Pad` or `Tablet tool`.

Additional properties only present in per-device configs:

```
enabled -> (only for mice / touchpads / touchdevices / keyboards) enables /  
disables the device (connects / disconnects from the on-screen cursor) -  
default: Enabled
```

Example config section:

```
device:royuan-akko-multi-modes-keyboard-b {  
    repeat_rate=50  
    repeat_delay=500  
    middle_button_emulation=0  
}
```

remember about the space after the end of the device's name (before the {)!

Info

Per-device layouts will not alter the keybind keymap, so for example with a global keymap of `us` and a per-device one of `fr`, the keybinds will still act as if you were on `us`.

Wallpapers

The hyprland background you see when you first start Hyprland is **NOT A WALLPAPER**, it's the default image rendered at the bottom of the render stack.

To set a wallpaper, use a wallpaper utility like `hyprpaper` or `swaybg`.

More can be found in Useful Utilities.

Blurring layerSurfaces

LayerSurfaces are not windows. These are for example: Your wallpapers, notification overlays, bars, etc.

If you really want to blur them, use a layerrule:

```
layerrule = blur,NAMESPACE  
# or  
layerrule = blur,address:0x<ADDRESS>
```

you can get the namespace / address from `hyprctl layers`.

To remove a layer rule (useful in dynamic situations) use:

```
layerrule = unset,<whatever you used before>
```

For example:

```
layerrule = unset,NAMESPACE
```

Setting the environment

Note

The `env` keyword works just like `exec-once`, meaning it will only fire once on Hyprland's launch.

You can use the `env` keyword to set environment variables at Hyprland's start, e.g.:

```
env = XCURSOR_SIZE, 24
```

You can also add a `d` flag if you want the env var to be exported to D-Bus (systemd only)

```
envd = XCURSOR_SIZE, 24
```

Important

Hyprland puts the raw string to the envvar. You should *not* add quotes around the values.

e.g.:

```
env = QT_QPA_PLATFORM, wayland
```

and **NOT**

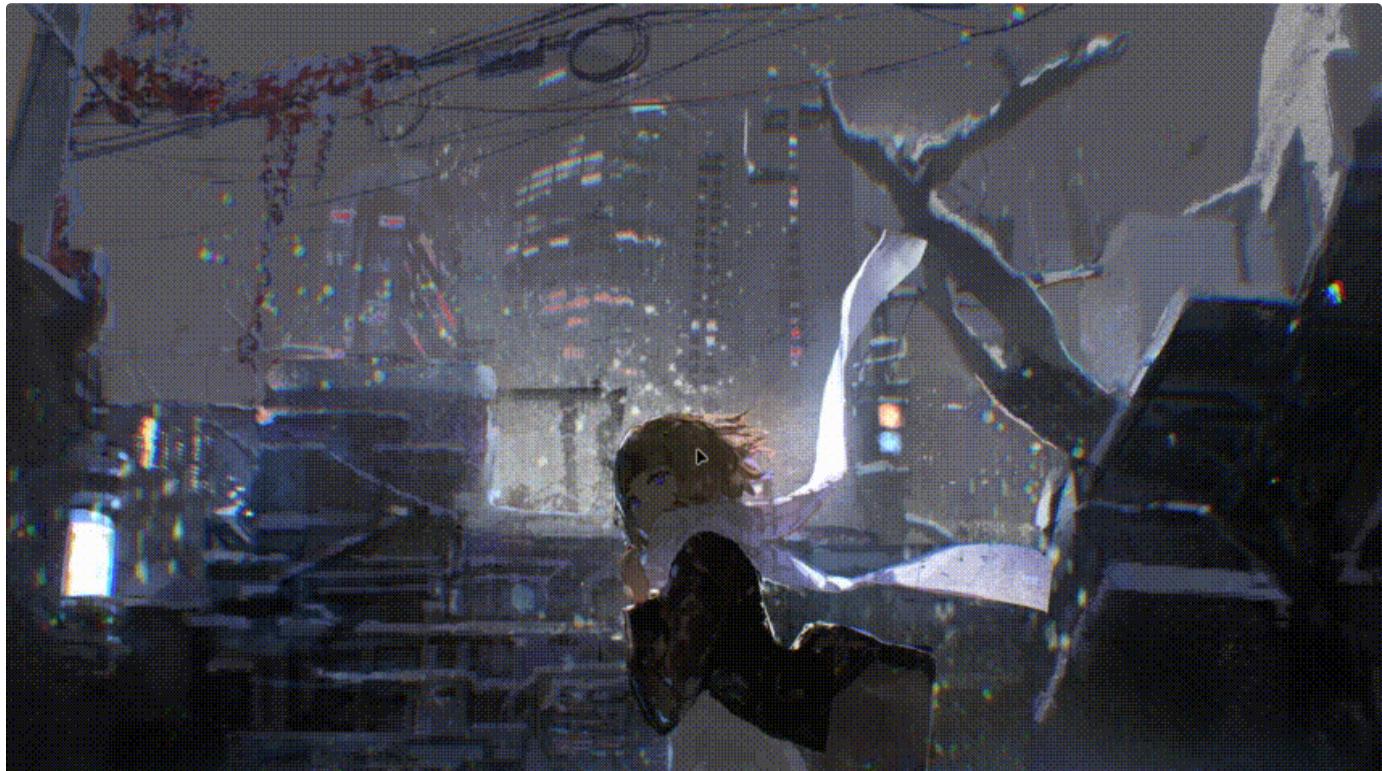
```
env = QT_QPA_PLATFORM, "wayland"
```

Hyprland Wiki

/Configuring/Master Layout

Master Layout

The master layout makes one (or more) window(s) be the “master”, taking (by default) the left part of the screen, and tiles the rest on the right. You can change the orientation on per-workspace basis if you want to use anything other than the default left/right split.



Config

category name master

| name | description | type | default |
|-------------------|---|-------------|----------------|
| allow_small_split | enable adding additional master windows in a horizontal split style | bool | false |

| name | description | type | default |
|----------------------|---|-------------|----------------|
| special_scale_factor | the scale of the special workspace windows. [0.0 - 1.0] | float | 1 |
| mfact | master split factor, the ratio of master split [0.0 - 1.0] | float | 0.55 |
| new_is_master | whether a newly open window should replace the master or join the slaves. | bool | true |
| new_on_top | whether a newly open window should be on the top of the stack | bool | false |
| no_gaps_when_only | whether to apply gaps when there is only one window on a workspace, aka. smart gaps. (default: disabled - 0) no border - 1, with border - 2 [0/1/2] | int | 0 |
| orientation | default placement of the master area, can be left, right, top, bottom or center | string | left |
| inherit_fullscreen | inherit fullscreen status when cycling/swapping to another window (e.g. monocle layout) | bool | true |
| always_center_master | when using orientation=center, keep the master window centered, even when it is the only window in the workspace. | bool | false |
| smart_resizing | if enabled, resizing direction will be determined by the mouse's position on the window (nearest to which corner). Else, it is based on the window's tiling position. | bool | true |
| drop_at_cursor | when enabled, dragging and dropping windows will put them at the cursor position. Otherwise, when dropped at the stack side, they will go to the | bool | true |

| name | description | type | default |
|-------------|--|-------------|----------------|
| | top/bottom of the stack depending on new_on_top. | | |

Dispatchers

layoutmsg commands:

| command | description | params |
|----------------|---|---|
| swapwithmaster | swaps the current window with master. If the current window is the master, swaps it with the first child. | either master (new focus is the new master window), child (new focus is the new child) or auto (which is the default, keeps the focus of the previously focused window) |
| focusmaster | focuses the master window. | either master (focus stays at master, even if it was selected before) or auto (which is the default, if the current window is the master, focuses the first child.) |
| cyclenext | focuses the next window respecting the layout | none |
| cycleprev | focuses the previous window respecting the layout | none |
| swapnext | swaps the focused window with the next window respecting the layout | none |
| swapprev | swaps the focused | none |

| command | description | params |
|-------------------|--|--------|
| | window with the previous window respecting the layout | |
| addmaster | adds a master to the master side. That will be the active window, if it's not a master, or the first non-master window. | none |
| removemaster | removes a master from the master side. That will be the active window, if it's a master, or the last master window. | none |
| orientationleft | sets the orientation for the current workspace to left (master area left, slave windows to the right, vertically stacked) | none |
| orientationright | sets the orientation for the current workspace to right (master area right, slave windows to the left, vertically stacked) | none |
| orientationtop | sets the orientation for the current workspace to top (master area top, slave windows to the bottom, horizontally stacked) | none |
| orientationbottom | sets the orientation for the current workspace to bottom (master area | none |

| command | description | params |
|-------------------|--|--|
| | bottom, slave windows to the top, horizontally stacked) | |
| orientationcenter | sets the orientation for the current workspace to center (master area center, slave windows alternate to the left and right, vertically stacked) | none |
| orientationnext | cycle to the next orientation for the current workspace (clockwise) | none |
| orientationprev | cycle to the previous orientation for the current workspace (counter-clockwise) | none |
| orientationcycle | cycle to the next orientation from the provided list, for the current workspace | allowed values: <code>left</code> , <code>top</code> , <code>right</code> , <code>bottom</code> , or <code>center</code> . The values have to be separated by a space. If left empty, it will work like <code>orientationnext</code> |
| mfact | change mfact, the master split ratio | the new split ratio, a float between 0.0 and 1.0 |
| rollnext | rotate the next window in stack to be the master, while keeping the focus on master | none |

| command | description | params |
|----------|---|--------|
| rollprev | rotate the previous window in stack to be the master, while keeping the focus on master | none |

params for the commands are separated by a single space

Info

example usage:

```
bind=MOD,KEY,layoutmsg,cyclenext
# behaves like xmonads promote feature (https://hackage.haskell.org/package/xmonad-contrib-0.17.1/docs/XMonad-Actions-Promote.html)
bind=MOD,KEY,layoutmsg,swapwithmaster master
```

Workspace Rules

layoutopt rules:

| rule | description | type |
|-----------------|--|--------|
| orientation:[o] | Sets the orientation of a workspace. For available orientations, see Config->orientation | string |

Example usage:

```
workspace = 2, layoutopt:orientation:top
```



 /Getting Started/Master Tutorial

Master Tutorial

If you are coming to Hyprland for the first time, this is the main tutorial to read.

Due to a lot of people doing stupid stuff, this tutorial will cover literally everything you need to just get things going. It does link to other pages where necessary.

- Install Hyprland
- NVIDIA?
- VM?
- Launching Hyprland
- In Hyprland
- Critical software
- Monitors config
- Apps / X11 replacements
- Fully configure
- Cursors
- Themes
- Force apps to use Wayland

Install Hyprland

See Installation and come back here once you have successfully installed Hyprland.

Install `kitty` (default terminal emulator) terminal. This is available in most distros' repositories.

NVIDIA?

If not using an NVIDIA card, skip this step

Please take a look at The Nvidia page before launching. It has a lot of info regarding the needed environment and tweaks.

VM?

If not using a VM, skip this step

In a VM, make sure you have 3D acceleration enabled in your virtio config (or virt-manager) otherwise Hyprland **will not work**.

You can also passthru a GPU to make it work.

Please bear in mind 3D accel in VMs may be pretty slow.

Launching Hyprland

Now, you can just execute `Hyprland` in your tty.

!IMPORTANT: Do **not** launch Hyprland with `root` permissions (don't `sudo`)

You can see some launch flags by doing `Hyprland -h`, these include setting the config path, ignoring a check for the above, etc.

Login managers are not officially supported, but here's a short compatibility list:

- SDDM → Works flawlessly. Install `sddm` $\geq 0.20.0$ or the latest git version (or `sddm-git` from AUR) to prevent SDDM bug 1476 (90s shutdowns).
- GDM → Works with the caveat of crashing Hyprland on the first launch
- ly → Works poorly

In Hyprland

You're good to go with your adventure, technically.

Use SUPER + Q to launch kitty. If you wish to choose the default terminal before you proceed, you can do so in `~/.config/hypr/hyprland.conf` (example config).

If you want the best experience with less hassle googling, keep reading.

Critical software

See the Must-have Software page for the crucial things to make Wayland / Hyprland / other apps work correctly.

Monitors config

See Configuring Hyprland page to learn all about configuring your displays.

Apps / X11 replacements

See the Useful Utilities page and the Sway wiki page just about that. You can also visit the Awesome-Hyprland repository for a more comprehensive list.

Fully configure

Head onto the Configuring Hyprland page to learn all about configuring Hyprland to your likings.

Cursors

Cursors are a notorious pain to set up when you don't know how. See this FAQ entry

If your cursor does not appear, then see this FAQ entry

Themes

Since this is not a full fledged Desktop Environment, you will need to use tools such as `lxappearance` and `nwg-look` (recommended) for GTK, and `qt5ct` / `qt6ct` for their respective Qt versions. Some older applications may also require `qt4ct`.

Force apps to use Wayland

A lot of apps will use Wayland by default. Chromium (and other browsers based on it or electron) don't. You need to pass `--enable-features=UseOzonePlatform --ozone-platform=wayland` to them or use `.conf` files where possible. Chromium-based browsers also should have a toggle in `chrome://flags`. Search for "ozone" and select Wayland.

For most electron apps, you should put the above in `~/.config/electron-flags.conf`. VSCode is known to not work with that though.

A few more environment variables for forcing Wayland mode are documented here.

You can check whether an app is running in xwayland or not with `hyprctl clients`.

The Hyprland Wiki, built with Hugo.



/Configuring/Monitors

Monitors

Table of contents

- Table of contents
- General
 - Custom modelines
 - Disabling a monitor
- Custom reserved area
- Extra args
 - Mirrored displays
 - 10 bit support
 - VRR
- Rotating
- Default workspace
 - Binding workspaces to a monitor

General

The general config of a monitor looks like this

```
monitor=name,resolution,position,scale
```

A common example:

```
monitor=DP-1,1920x1080@144,0x0,1
```

will tell Hyprland to make the monitor on DP-1 a 1920x1080 display, at 144Hz, 0x0 off from the top left corner, with a scale of 1 (unscaled).

To list all available monitors (active and inactive):

```
hyprctl monitors all
```

Monitors are positioned on a virtual “layout”. The `position` is the position of said display in the layout. (calculated from the top-left corner)

For example:

```
monitor=DP-1, 1920x1080, 0x0, 1  
monitor=DP-2, 1920x1080, 1920x0, 1
```

will tell hyprland to make DP-1 on the *left* of DP-2, while

```
monitor=DP-1, 1920x1080, 1920x0, 1  
monitor=DP-2, 1920x1080, 0x0, 1
```

will tell hyprland to make DP-1 on the *right*.

The `position` may contain *negative* values, so the above example could also be written as

```
monitor=DP-1, 1920x1080, 0x0, 1  
monitor=DP-2, 1920x1080, -1920x0, 1
```

Tip

The position is calculated with the scaled (and transformed) resolution, meaning if you want your 4K monitor with scale 2 to the left of your 1080p one, you'd use the position `1920x0` for the second screen. ($3840 / 2$) If the monitor is also rotated 90 degrees (vertical), you'd use `1080x0`.

Leaving the name empty will define a fallback rule to use when no other rules match.

You can use `preferred` as a resolution to use the display's preferred size and `auto` as a position to let Hyprland decide on a position for you.

You can also use `auto` as a scale to let Hyprland decide on a scale for you. These depend on the PPI of the monitor.

Recommended rule for quickly plugging in random monitors:

```
monitor=,preferred,auto,1
```

Will make any monitor that was not specified with an explicit rule automatically placed on the right of the other(s) with its preferred resolution.

Alternatively, you can use the `highres` or `highrr` rules in order to get the best possible resolution or refreshrate mix.

for a focus on refreshrate use this:

```
monitor=,highrr,auto,1
```

for a focus on resolution this:

```
monitor=,highres,auto,1
```

For more specific rules, you can also use the output's description (see `hyprctl monitors` for more details). If the output of `hyprctl monitors` looks like the following:

```
Monitor eDP-1 (ID 0):
    1920x1080@60.00100 at 0x0
    description: Chimei Innolux Corporation 0x150C (eDP-1)
    make: Chimei Innolux Corporation
    model: 0x150C
    [...]
```

then the `description` value up to the portname (`eDP-1`) can be used to specify the monitor:

```
monitor=desc:Chimei Innolux Corporation 0x150C,preferred,auto,1.5
```

Remember to remove the `(portname)` !

Custom modelines

You can set up a custom modeline by changing the resolution field to a modeline, for example:

```
monitor = DP-1, modeline 1071.101 3840 3848 3880 3920 2160 2263 2271 2277
```

```
+hsync -vsync, 0x0, 1
```

Disabling a monitor

To disable a monitor, use

```
monitor=name, disable
```

Tip

Disabling a monitor will literally remove it from the layout, moving all windows and workspaces to any remaining ones. If you want to disable your monitor in a screensaver style (just turn off the monitor) use the `dpm` dispatcher.

Custom reserved area

If your workflow requires custom reserved area, you can add it with

```
monitor=name, addreserved, TOP, BOTTOM, LEFT, RIGHT
```

Where `TOP` `BOTTOM` `LEFT` `RIGHT` are integers in pixels of the reserved area to add. This does stack on top of the calculated one, (e.g. bars) but you may only use one of these rules per monitor in the config.

Extra args

You can combine extra arguments at the end of the monitor rule, examples:

```
monitor=eDP-1, 2880x1800@90, 0x0, 1, transform, 1, mirror, DP-2, bitdepth, 10
```

See below for more detail about each argument.

Mirrored displays

If you want to mirror a display, add a `,mirror, [NAME]` at the end of the monitor rule, examples:

```
monitor=DP-3,1920x1080@60,0x0,1,mirror,DP-2  
monitor=,preferred,auto,1,mirror,DP-1
```

Please remember that mirroring displays will not “re-render” everything for your second monitor, so if mirroring a 1080p screen onto a 4K one, the resolution will still be 1080p on the 4K display. This also means squishing and stretching will occur on non-matching resolutions.

10 bit support

If you want to enable 10 bit support for your display, add a `,bitdepth,10` at the end of the monitor rule, e.g.:

```
monitor=eDP-1,2880x1800@90,0x0,1,bitdepth,10
```

NOTE Colors registered in Hyprland (e.g. the border color) do not support 10 bit.

NOTE Some applications do not support screen capture with 10 bit enabled.

VRR

Per-display VRR can be done by adding `,vrr,x` where `x` is the mode from the variables page.

Rotating

If you want to rotate a monitor, add a `,transform,x` at the end of the monitor rule, where `x` corresponds to a transform number, e.g.:

```
monitor=eDP-1,2880x1800@90,0x0,1,transform,1
```

Transform list:

```
normal (no transforms) -> 0  
90 degrees -> 1  
180 degrees -> 2  
270 degrees -> 3  
flipped -> 4  
flipped + 90 degrees -> 5  
flipped + 180 degrees -> 6
```

flipped + 270 degrees -> 7

Note

If you're using a touchscreen, you'll also have to rotate its digitizer to match:

```
input {  
    touchdevice {  
        transform = 1  
    }  
}
```

This will be done automatically when #3544 lands.

Default workspace

See Workspace Rules.

Binding workspaces to a monitor

See Workspace Rules.

The Hyprland Wiki, built with Hugo.

Hyprland Wiki

 /Useful Utilities/Must Have

Must Have

This page documents software that is critical / very important to have running for a smooth Wayland / Hyprland experience.

DEs like KDE / Gnome will do this automatically, Hyprland will not (because you might want to use something else)

A notification daemon

Starting method: most likely manual (`exec-once`)

Many apps (e.g. Discord) may freeze without one running.

Use e.g. `dunst` , `mako` , `swaync` , etc.

Pipewire

Starting method: automatic

Pipewire is not necessarily required, but screensharing will not work without it.

Install `pipewire` and `wireplumber` (**not** `pipewire-media-session`)

Xdg Desktop Portal

Starting method: Automatic on Systemd, manual otherwise

XDG Desktop Portal handles a lot of stuff for your desktop, like file pickers, screensharing, etc.

See The Hyprland Desktop Portal Page

Authentication Agent

Starting method: manual (exec-once)

Authentication agents are the things that pop up a window asking you for a password whenever an app wants to elevate its privileges.

Our recommendation is the KDE one. For arch, it's `polkit-kde-agent`.

You can autostart it with `exec-once=/usr/lib/polkit-kde-authentication-agent-1`. On some distributions you might have to use a different path `/usr/libexec/polkit-kde-authentication-agent-1`.

On other distributions that use a more recent version, such as Gentoo, it may be necessary to use `exec-once=/usr/lib64/libexec/polkit-kde-authentication-agent-1` instead.

Qt Wayland Support

Starting method: none (just a library)

Install `qt5-wayland` and `qt6-wayland`.

The Hyprland Wiki, built with Hugo.



/Nix/Options & Overrides

Options & Overrides

You can override the package through `.override` or `.overrideAttrs`. This is easily achievable on NixOS or Home Manager.

Package options

These are the default options that the Hyprland package is built with. These can be changed by setting the appropriate option to `true` / `false`.

Package

```
(pkgs.hyprland.override { # or
  inputs.hyprland.packages.${pkgs.system}.hyprland
  enableXWayland = true; # whether to enable XWayland
  legacyRenderer = false; # whether to use the legacy renderer (for old GPUs)
  withSystemd = true;     # whether to build with systemd support
})
```

NixOS & HM modules

```
programs.hyprland = { # or wayland.windowManager.hyprland
  enable = true;
  xwayland.enable = true;
};
```

Options descriptions

XWayland

XWayland is enabled by default in the Nix package. You can disable it either in the package itself, or through the module options.

XWayland HiDPI

See XWayland.

Using Nix repl

If you're using Nix (and not NixOS or Home Manager) and you want to override, you can do it like this

```
$ nix repl
nix-repl> :lf "github:hyprwm/Hyprland"
nix-repl> :bl outputs.packages.x86_64-linux.hyprland.override { /* flag here
*/ }
```

Then you can run Hyprland from the built path.

The Hyprland Wiki, built with Hugo.

Hyprland Wiki

 /Useful Utilities/Other

Other

Here you will find links to some other projects that may not fit into any of the above categories.

Workspace management

hyprsome by *sopa0*: Awesome-like workspaces for Hyprland.

Keyboard layout management

hyprland-per-window-layout by *MahouShoujoMivutilde* and *coffebar*: Per window keyboard layouts for Hyprland.

IPC wrappers

hyprland-rs by *yavko*: A neat wrapper for Hyprland's IPC written in Rust

Screen shaders/color temperature

hyprshade by *loqusion*: Utility for swapping and scheduling screen shaders; also functions as an automatic color temperature shifter. (Useful for Nvidia users for whom other color temperature shifting apps do not work.)

Automatically Mounting Using `udiskie`

Starting method: manual ('exec-once')

USB Mass storage devices, like thumb drives, mobile phones, digital cameras, etc. do not mount automatically to the file system.

We generally have to manually mount it, often using root and `umount` to do so.

Many popular DEs automatically handle this by using `udisks2` wrappers.

`udiskie` is a `udisks2` front-end that allows to manage removable media such as CDs or flash drives from userspace.

Install `udiskie` via your repositories, or build manually

Head over to your `~/.config/hypr/hyprland.conf` and add the following lines:

```
exec-once = udiskie &
```

What this does is launches `udiskie` and `&` argument launches it in the background.

See more uses here .

The Hyprland Wiki, built with Hugo.



/Configuring/Performance

Performance

This page documents known tricks and fixes to boost performance if by any chance you stumble upon problems or you do not care that much about animations.

Fractional scaling

Wayland fractional scaling is a lot better than before, but it is not perfect. Some applications do not support it yet or the support is experimental at best. If you have problems with your graphics card having high usage and hyprland feeling laggy then try setting the scaling to integer numbers such as `1` or `2` like in this example `monitor=,preferred,auto,2`.

Low FPS/stutter/FPS drops on Intel iGPU with TLP (mainly laptops)

The TLP defaults are rather aggressive, setting `INTEL_GPU_MIN_FREQ_ON_AC` and/or `INTEL_GPU_MIN_FREQ_ON_BAT` in `/etc/tlp.conf` to something slightly higher (e.g. to 500 from 300) will reduce stutter significantly or, in the best case, remove it completely.

How do I make Hyprland draw as little power as possible on my laptop?

Useful Optimizations:

- `decoration:blur = false` and `decoration:drop_shadow = false` to disable fancy but battery hungry effects.
- `misc:vfr = true`, since it'll lower the amount of sent frames when nothing is happening on-screen.

My games work poorly, especially proton ones

Use `gamescope`, tends to fix any and all issues with wayland/Hyprland.

The Hyprland Wiki, built with Hugo.



[/Plugins/Plugin Guidelines](#)

Plugin Guidelines

This page documents the recommended guidelines for making a stable and neat plugin.

- Making your plugin compatible with hyprpm
 - Repository metadata
 - Plugins
 - Commit pins
- Formatting
- Usage of the API
- Function Hooks
- Threads

Making your plugin compatible with hyprpm

In order for your plugin to be installable by `hyprpm`, you need a manifest.

`hyprpm` will parse `hyprload` manifests just fine, but it's recommended to use the more powerful `hyprpm` manifest.

Make a file in the root of your repository called `hyprpm.toml`.

Repository metadata

At the beginning, put some metadata about your plugin:

```
[repository]  
name = "MyPlugin"  
authors = ["Me"]  
commit_pins = [  
    "3bb9c7c5cf4f2ee30bf821501499f2308d616f94",  
    "efee74a7404495dbda70205824d6e9fc923ccdae"],  
    "d74607e414dcd16911089a6d4b6aeb661c880923",
```

```
"efee74a7404495dbda70205824d6e9fc923ccdae"]  
]
```

name and authors are required. commit_pins are optional. See commit pins for more info.

Plugins

For each plugin, make a category like this:

```
[plugin-name]  
description = "An epic plugin that will change the world!"  
authors = ["Me"]  
output = "plugin.so"  
build = [  
    "make all"  
]
```

description , authors are optional. output and build are required.

build are the commands that hyprpm will run in the root of the repo to build the plugin. Every command will reset the cwd to the repo root.

output is the path to the output .so file from the root of the repo.

Commit pins

Commit pins allow you to manage versioning of your plugin. they are pairs of hash,hash , where the first hash is the Hyprland commit hash, and the second is your plugin's corresponding commit hash.

For example, in the manifest above, d74607e414dcd16911089a6d4b6aeb661c880923 corresponds to Hyprland's 0.33.1 release, which means that if someone is running 0.33.1 , hyprpm will reset your plugin to commit hash efee74a7404495dbda70205824d6e9fc923ccdae .

It's recommended you add a pin for each Hyprland release. If no pin matches, latest git will be used.

Formatting

Although Hyprland plugins obviously are not *required* to follow Hyprland's formatting, naming

conventions, etc. it might be a good idea to keep your code consistent. See `.clang-format` in the Hyprland repo.

Usage of the API

It's always advised to use the API entries whenever possible, as they are guaranteed stability as long as the version matches.

It is, of course, possible to use the internal methods by just including the proper headers, but it should not be treated as the default way of doing things.

Hyprland's internal methods may be changed, removed or added without any prior notice. It is worth noting though that methods that "seem" fundamental, like e.g. `focusWindow` or `mouseMoveUnified` probably are, and are unlikely to change their general method of functioning.

Function Hooks

Function hooks allow your plugin to intercept all calls to a function of your choice. They are to be treated as a last resort, as they are the easiest thing to break between updates.

Always prefer using Event Hooks.

Threads

The Wayland event loop is strictly single-threaded. It is not recommended to create threads in your code, unless they are fully detached from the Hyprland process. (e.g. saving a file)

The Hyprland Wiki, built with Hugo.



/Useful Utilities/Screen Sharing

Screen Sharing

Screensharing is done through PipeWire on Wayland.

Prerequisites

Make sure you have `pipewire` and `wireplumber` installed, enabled and running if you don't have them yet.

Screensharing

Read this amazing gist by PowerBall253 for a great tutorial.

Better screensharing

See the hyprland portal page

XWayland

If your screensharing application is running under XWayland (like Discord, Skype,...), it can only see other XWayland windows and cannot share an entire screen or a Wayland window.

The KDE-team has implemented a workaround for this called `xwaylandvideobridge`. You can use this AUR package on ArchLinux. Note that Hyprland currently doesn't support the way it tries to hide the main window, so you will have to create some window-rules to achieve the same effect. See this issue for more information. For example:

```
windowrulev2 = opacity 0.0 override 0.0 override, class:^(xwaylandvideobridge)$  
windowrulev2 = noanim, class:^(xwaylandvideobridge)$  
windowrulev2 = noinitialfocus, class:^(xwaylandvideobridge)$  
windowrulev2 = maxsize 1 1, class:^(xwaylandvideobridge)$  
windowrulev2 = noblur, class:^(xwaylandvideobridge)$
```

The Hyprland Wiki, built with Hugo.



/Useful Utilities/Status Bars

Status Bars

Table of contents

- Table of contents
- Waybar
 - How to launch
 - Waybar popups render behind the windows
 - Active workspace doesn't show up
 - Scrolling through workspaces
 - Clicking on a workspace icon does not work!
 - Window title is missing
- Eww
 - Configuration
 - `~/.config/eww.yuck`
 - `~/.config/eww/scripts/change-active-workspace`
 - `~/.config/eww/scripts/get-active-workspace`
 - `~/.config/eww/scripts/get-workspaces`
 - `~/.config/eww/eww.yuck`
 - `~/.config/eww/scripts/get-window-title`
- Hybrid
 - Configuration
 - How to launch
 - Blur

Waybar

Waybar is a GTK status bar made specifically for wlroots compositors and supports Hyprland

by default. To use it, it's recommended to use your distro's package.

If you want to use the workspaces module, first, copy the configuration files from `/etc/xdg/waybar/` into `~/.config/waybar/`. Then, in `~/.config/waybar/config` replace all the references to `sway/workspaces` with `hyprland/workspaces`.

For more info regarding configuration, see [The Waybar Wiki](#).

How to launch

After getting everything set up, you might want to check if Waybar is configured to your liking. To launch it, simply type `waybar` into your terminal. If you would like waybar to launch alongside hyprland, you can do this by adding a line to your hyprland configuration that reads `exec-once=waybar`

Waybar popups render behind the windows

In `~/.config/waybar/config`, make sure that you have the `layer` configuration set to `top` and not `bottom`.

Active workspace doesn't show up

Replace `#workspaces button.focused` with `#workspaces button.active` in `~/.config/waybar/style.css`.

Scrolling through workspaces

Since there a lot of configuration options from `sway/workspaces` are missing, you should deduce some of them by yourself. In the case of scrolling, it should look like this:

```
"hyprland/workspaces": {  
    "format": "{icon}",  
    "on-scroll-up": "hyprctl dispatch workspace e+1",  
    "on-scroll-down": "hyprctl dispatch workspace e-1"  
}
```

Clicking on a workspace icon does not work!

On the `hyprland/workspaces` module, add `"on-click": "activate"`. That's the purpose

of the `sed` command used before building Waybar: the default way to select a workspace by clicking uses the `swaymsg`'s way, and thus it is required to edit this function to make it work with `hyprctl`.

Window title is missing

Follow the above instructions to make sure everything is working. The prefix for the window module that provides the title is `hyprland` not `wlr`. In your waybar config, insert this module:

```
"modules-center": ["hyprland/window"],
```

If you are using a multiple monitors, you may want to also insert this module configuration:

```
"hyprland/window": {  
    "max-length": 200,  
    "separate-outputs": true  
},
```

Eww

In order to use Eww, you first have to install it, either using your distro's package manager, by searching `eww-wayland`, or by manually compiling. In the latter case, you can follow the instructions.

Configuration

After you've successfully installed Eww, you can move onto configuring it. There are a few examples listed in the Readme. It's also highly recommended to read through the Configuration options.

Important

Read the Wayland section carefully before asking why your bar doesn't work.

Here are some example widgets that might be useful for Hyprland:

- ▶ Workspaces widget
- ▶ Active window title widget

Hybrid

Like Waybar, Hybrid is a GTK status bar mainly focused for wlroots compositors.

You can install it from the AUR by the name `hybrid-bar`.

Configuration

The configuration is done through JSON, more information is available [here](#).

How to launch

After configuring HybridBar, you can launch it by typing `hybrid-bar` into your terminal to try it out. It is also possible to set it to launch at start, to do this you can add a line to your hyprland configuration that reads `exec-once=hybrid-bar`

Blur

To activate blur, set `blurls=NAMESPACE` in your hyprland configuration, where `NAMESPACE` is the gtk-layer-shell namespace of your HybridBar. The default namespace is `gtk-layer-shell` and can be changed in the HybridBar configuration at

```
{  
    "hybrid": {  
        "namespace": "namespace name"  
    }  
}
```

The Hyprland Wiki, built with Hugo.



/Configuring/Tearing

Tearing

Screen tearing is used to reduce latency and/or jitter in games.

Enabling tearing

To enable tearing:

- Set `general:allow_tearing to true`. This is a “master toggle”
- Add `env = WLR_DRM_NO_ATOMIC,1` to your Hyprland config. This disables the usage of a newer kernel DRM API that doesn’t support tearing yet.
- Add an `immediate` windowrule to your game of choice. This makes sure that Hyprland will tear it.

Note

Please note that tearing will only be in effect when the game is in fullscreen and the only thing visible on the screen.

Example snippet:

```
general {  
    allow_tearing = true  
}  
  
env = WLR_DRM_NO_ATOMIC,1  
  
windowrulev2 = immediate, class:^(cs2)$
```

Note

`env = WLR_DRM_NO_ATOMIC,1` is not recommended. If your kernel ver is ≥ 6.8 , you can remove it.

For kernels < 6.8, this env is required.

Check your kernel version with `uname -r`.

Warning

If you experience graphical issues, you may be out of luck. Tearing support is experimental.

See the likely culprits below.

Common issues

No tearing at all

Make sure your windowrules are matching and you have the master toggle enabled.

Also make sure nothing except for your game is showing on your monitor. No notifications, overlays, lockscreens, bars, other windows, etc. (on a different monitor is fine)

Apps that should tear, freeze

Almost definitely means your GPU driver does not support tearing, like e.g. Intel's.

Please *do not* report issues if this is the culprit.

Graphical artifacts (random colorful pixels, etc)

Likely issue with your graphics driver.

Please *do not* report issues if this is the culprit. Unfortunately, it's most likely your GPU driver's fault.

Could be the below as well

Other graphical issues

or

Hyprland instantly crashes on launch

Likely issue with `WLR_DRM_NO_ATOMIC`.

NO_ATOMIC forces the use of a legacy, less tested drm API.

Please *do not* report issues if this is the culprit. Unfortunately, you will have to wait for the Linux kernel to support tearing page flips on the atomic API.

The Hyprland Wiki, built with Hugo.

sddm /
sddm[Code](#)[Issues 513](#)[Pull requests 42](#)[Actions](#)[Projects 2](#)[Wiki](#)

Theming

[Jump to bottom](#)Fabian Vogt edited this page on Jul 27, 2023 · 7 revisions

Themes

SDDM themes are created using QtQuick framework, a declarative framework to develop next-generation, hardware-accelerated user interfaces with fluid animations. QtQuick offers some basic components

On top of QtQuick, we provide some custom components to make theme development even easier. For example a picturebox which can show user avatars. Most of the components can be used as views in a model-view sense.

We also provide models containing information about the screens, sessions available and users. Connect these with the provided components and you have fully working solution. For example, below is the whole *code* needed to create a session selection combobox:

```
ComboBox {  
    id: session  
    arrowIcon: "angle-down.png"  
    model: sessionModel  
    index: sessionModel.lastIndex  
}
```



Proxy Object

We provide a proxy object, called as `sddm` to the themes as a context property. This object holds some useful properties about the host system. It also acts as a proxy between the greeter and the daemon. All of the methods called on this object will be transferred to the daemon through a local socket and will be executed there.

Properties

hostname: Holds the name of the host computer.

canPowerOff: true, if we can power off the machine; false, otherwise

canReboot: true, if we can reboot the machine; false, otherwise

canSuspend: true, if the machine supports suspending to the memory; false, otherwise

canHibernate: true, if the machine supports hibernating, e.g suspending to the disk; false, otherwise

~~**canHybridSleep:** true, if the machine supports hybrid sleep, e.g suspending to both~~

canHybridSleep. true, if the machine supports hybrid sleep, e.g suspending to both memory and disk; false, otherwise

Methods

powerOff(): Powers off the machine.

reboot(): Reboots the machine.

suspend(): Suspends the machine to the memory.

hibernate(): Suspends the machine to the disk.

hybridSleep(): Suspends the machine both to the memory and the disk.

login(user, password, sessionIndex): Attempts to login as the `user`, using the `password` into the session pointed by the `sessionIndex`. Either the `loginFailed` or the `loginSucceeded` signal will be emitted depending on whether the operation is successful or not.

Signals

loginFailed(): Emitted when a requested login operation fails.

loginSucceeded(): Emitted when a requested login operation succeeds.

Data Models

Besides the proxy object we offer a few models that can be hooked to the views to handle multiple screens or enable selection of users or sessions.

screenModel: This is a list model containing geometry information of the screens available. This model only provides logical screen numbers and geometries. If you have two physical monitors, but configured to be duplicates we only report one screen.

For each screen the model provides `name` and `geometry` properties. The model also provides, a `primary` property pointing to the index of the primary monitor and a `geometry` method which takes a monitor index and returns the geometry of it. If you `-1` to the `geometry` method it will return the united geometry of all the screens available.

sessionModel: This is a list model. Contains information about the desktop sessions installed on the system. This information is gathered by parsing the desktop files in the `/usr/share/xsessions`. These desktop files are generally installed when you install a

a desktop environment or a window manager.

For each session, the model provides `file`, `name`, `exec` and `comment` properties. Also there is a `lastIndex` property, pointing to the last session the user successfully logged in.

userModel: This is list model. Contains information about the users available on the system. This information is gathered by parsing the `/etc/passwd` file. To prevent system users polluting the user model we only show users with user ids greater than a certain threshold. This threshold is adjustable through the config file and called `MinimumUid`.

For each user the model provides `name`, `realName`, `homeDir` and `icon` properties. This model also has a `lastIndex` property holding the index of the last user successfully logged in, and a `lastUser` property containing the name of the last user successfully logged in.

Keyboard stuff

`keyboard` object.

keyboard.numLock - boolean, numlock state (can be changed).

keyboard.capsLock - boolean, capslock state (can be changed).

keyboard.currentLayout - integer, id of current layout (can be changed).

keyboard.layouts - list model, each element provides properties `shortName` - 2 symbol country code and `longName` - full country name

Localization

See [this](#)

Theme Configuration

Themes can have a configuration file to allow changing of appearance (e.g. wallpaper path and type) and behaviour (e.g. controls to show). This configuration file is in the theme's directory and by default called `theme.conf`. The filename can be changed by setting

`[SddmGreeterTheme] ConfigFile=foobar.ini` in `metadata.desktop`. To change configuration values, create a file next to it with the filename + a `.user` suffix, e.g. `theme.conf.user` and write the desired values into it. This split is such that theme authors, editors and packagers can specify default values and changes performed by users are not lost on upgrades.

Themes can access the configuration values through the `config` object in the global QML namespace.

`config.keys().indexOf(key)` Returns -1 if key is not specified in the configuration file(s).

Old implicitly typed API

The configuration value is directly accessible as member of the `config` object, e.g.

```
config.timeout for timeout=10 .
```

It's discouraged to use this API as all values are returned as `string` type, which means that for e.g. `doStuff=false` , `if(config.doStuff)` will be executed and for `timeout=10` , `config.timeout + 1` will be `101` !

New explicitly typed API (since SDDM 0.20)

`config.boolValue(key)` Returns the value of the setting as boolean, `false` if unset. This uses `QVariant::toBool` internally, so `false` , `0` or an empty string in the configuration file count as `false` , everything else is `true` . Example: `if(config.boolValue("doStuff")) doStuff(); .`

`config.intValue(key)` Returns the value of the setting as integer, `0` if unset or conversion fails. This uses `QVariant::toInt` internally. Example: `timeout_plus_one = config.intValue("timeout") + 1; .`

`config.realValue(key)` Returns the value of the setting as real, `0.0` if unset or conversion fails. This uses `QVariant::toReal` internally.

`config.stringValue(key)` Returns the value of the setting as string, an empty string if unset.

Testing

You can test your themes using `sddm-greeter` . Note that in this mode, actions like shutdown, suspend or login will have no effect.

```
sddm-greeter --theme /path/to/you/theme
```



▼ Pages 19

Find a page...

▶

[▶ \[DRAFT\] 1.0.0 Release Announcement](#)[▶ \[TEMPLATE\] Release announcement](#)[▶ Architecture](#)[▶ Authentication](#)[▶ Brno Tasks](#)[▶ Installation](#)[▶ Localization](#)

Show 4 more pages...

[Clone this wiki locally](#)<https://github.com/sddm/sddm.wiki.git>



/Configuring/Uncommon Tips & Tricks

Uncommon Tips & Tricks

Switchable keyboard layouts

The easiest way to accomplish this is to set this using XKB settings, for example:

```
input {  
    kb_layout = us,pl  
    kb_options = grp:alt_shift_toggle  
}
```

Important

The first layout defined in the input section will be the one used for binds.

For example: `us,ua` -> config binds would be e.g. `SUPER, A`, while on `ua,us` -> `SUPER, Cyrillic_ef`

You can also bind a key to execute `hyprctl switchxkblayout` for more keybind freedom.
See [Using hyprctl](#).

To find the valid layouts and `kb_options`, you can check out the `/usr/share/X11/xkb/rules/base.lst`. For example:

To get the layout name of a language:

```
grep -i 'persian' /usr/share/X11/xkb/rules/base.lst
```

To get the list of keyboard shortcuts you can put in the `kb_options` to toggle keyboard layouts:

```
grep 'grp:.+toggle' /usr/share/X11/xkb/rules/base.lst
```

Disabling keybinds with one master keybind

If you want to disable all keybinds with another keybind (make a keybind toggle of sorts) you can just use a submap with only a keybind to exit it.

```
bind=MOD,KEY, submap, clean
submap=clean
bind=MOD,KEY, submap, reset
submap=reset
```

Remap Caps-Lock to Ctrl

```
input {
    kb_options = ctrl:nocaps
}
```

Minimize Steam instead of killing

Steam will exit entirely when its last window is closed using the `killactive` dispatcher. To minimize Steam to tray, use the following script to close applications:

```
if [ "$(hyprctl activewindow -j | jq -r ".class")" = "Steam" ]; then
    xdotool getactivewindow windowunmap
else
    hyprctl dispatch killactive ""
fi
```

Window Dancing

Some XWayland games like Rhythm Doctor and Friday Night Funkin' mods like to move the windows by themselves, but that often doesn't work by default.

For example, if you want to configure Rhythm Doctor, you'd have to:

1. Set input rules

```
input {  
    # ...  
    follow_mouse=0  
    float_switch_override_focus=0  
}
```

2. Set the windowrule

```
windowrule=windowdance,title:^(Rhythm Doctor)$  
# windowrule=forceinput,title:^(Rhythm Doctor)$ # May also be needed
```

3. Have fun!

Click the GIF below to see a full demo video



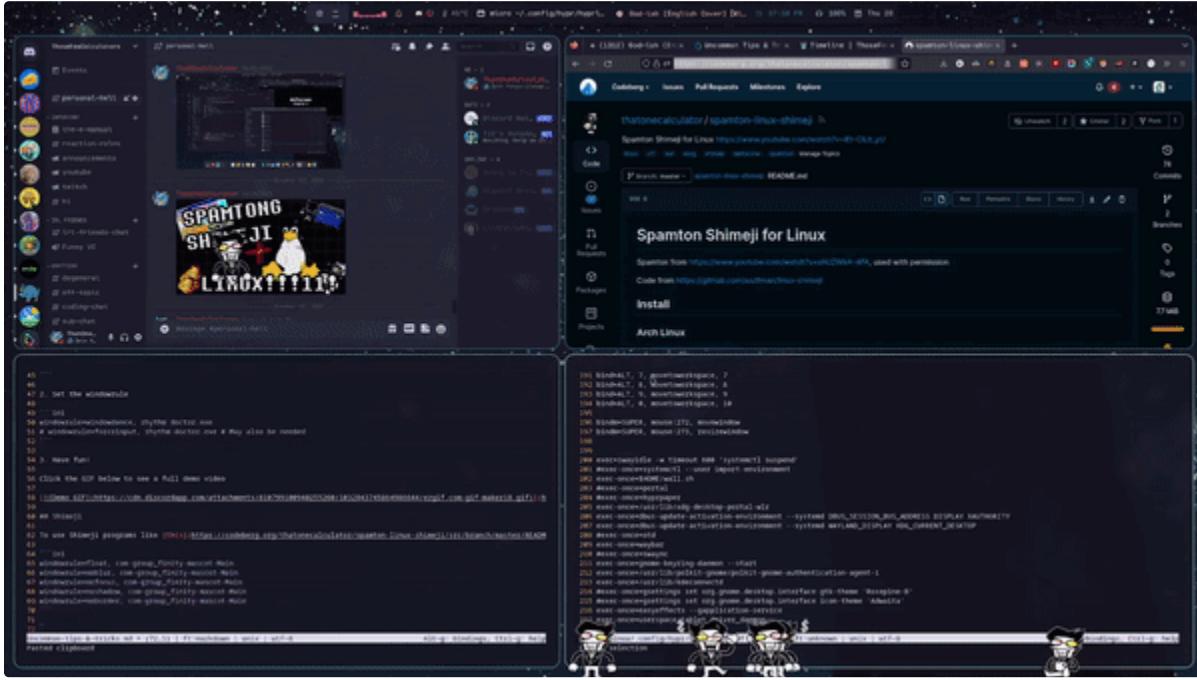
Shimeji

To use Shimeji programs like this, set the following rules:

```
windowrule=float, com-group_finity-mascot-Main  
windowrule=noblur, com-group_finity-mascot-Main  
windowrule=nofocus, com-group_finity-mascot-Main  
windowrule=noshadow, com-group_finity-mascot-Main  
windowrule=noborder, com-group_finity-mascot-Main
```

Note

The app indicator probably won't show, so you'll have to `killall -9 java` to kill them.



Toggle animations/blur/etc hotkey

For increased performance in games, or for less distractions at a keypress

1. create file `~/.config/hypr/gamemode.sh` `&& chmod +x ~/.config/hypr/gamemode.sh` and add:

```
#!/usr/bin/env sh
HYPRGAMEMODE=$(hyprctl getoption animations:enabled | awk 'NR==2{print $2}')
if [ "$HYPRGAMEMODE" = 1 ] ; then
    hyprctl --batch \"\
        keyword animations:enabled 0; \
        keyword decoration:drop_shadow 0; \
        keyword decoration:blur:enabled 0; \
        keyword general:gaps_in 0; \
        keyword general:gaps_out 0; \
        keyword general:border_size 1; \
        keyword decoration:rounding 0\""
    exit
fi
hyprctl reload
```

Edit to your liking of course. If animations are enabled, it disables all the pretty stuff. Otherwise, the script reloads your config to grab your defaults.

2. Add this to your `hyprland.conf`:

```
bind = WIN, F1, exec, ~/.config/hypr/gamemode.sh
```

The hotkey toggle will be WIN+F1, but you can change this to whatever you want.

The Hyprland Wiki, built with Hugo.



/Configuring/Using Hyprctl

Using Hyprctl

`hyprctl` is a utility for controlling some parts of the compositor from a CLI or a script. If you install with `make install`, or any package, it should automatically be installed.

To check if `hyprctl` is installed, simply execute it by issuing `hyprctl` in the terminal.

If it's not, go to the repo root and `./hyprctl`. Issue a `make all` and then `sudo cp ./hyprctl /usr/bin`.

Using Hyprctl

Warning

hyprctl calls will be dispatched by the compositor *synchronously*, meaning any spam of the utility will cause slowdowns. It's recommended to use `--batch` for many control calls, and limiting the amount of info calls.

For live event handling, see the `socket2`.

Commands

Dispatch

issue a `dispatch` to call a keybind dispatcher with an arg.

An arg has to be present, for dispatchers without parameters it can be anything.

To pass an argument starting with `-` or `--`, such as command line options to `exec` programs, pass `--` as an option. This will disable any subsequent parsing of options by `hyprctl`.

Examples:

```
hyprctl dispatch exec kitty  
hyprctl dispatch -- exec kitty --single-instance  
hyprctl dispatch pseudo x
```

Returns: ok on success, an error message on fail.

See Dispatchers for a list of dispatchers.

Keyword

issue a keyword to call a config keyword dynamically.

Examples:

```
hyprctl keyword bind SUPER,0,pseudo  
hyprctl keyword general:border_size 10  
hyprctl keyword monitor DP-3,1920x1080@144,0x0,1
```

Returns: ok on success, an error message on fail.

Reload

issue a reload to force reload the config.

kill

issue a kill to get into a kill mode, where you can kill an app by clicking on it. You can exit it with ESCAPE.

Kind of like xkill.

setcursor

Sets the cursor theme and reloads the cursor manager. Will set the theme for everything except GTK, because GTK.

params: theme and size

e.g.:

```
hyprctl setcursor Bibata-Modern-Classic 24
```

output

Allows you to add and remove fake outputs to your preferred backend.

params: `create` or `remove` and `backend` or `name` respectively.

For `create`:

pass a backend name: `wayland` , `x11` , `headless` or `auto` . On a *real* hyprland session, if you're looking for a VNC / RDP type thing, it's 99% going to be `headless` .

For `remove`:

pass the output's name, as found in `hyprctl monitors` . Please be aware you are *not* allowed to remove real displays with this command.

e.g.:

```
# will create a 1920x1080 headless display, for example to use with RDP.  
hyprctl output create headless  
  
# will remove the above display, provided its name was HEADLESS-1  
hyprctl output remove HEADLESS-1
```

switchxkblayout

Sets the xkb layout index for a keyboard.

For example, if you set:

```
device:my-epic-keyboard-v1 {  
    kb_layout=us,p1,de  
}
```

You can use this command to switch between them.

```
hyprctl switchxkblayout [DEVICE] [CMD]
```

where `CMD` is either `next` for next, `prev` for previous, or `ID` for a specific one (in the above case, `us : 0, p1 : 1, de : 2`). You can find the `DEVICE` using `hyprctl devices`

command.

example command for a typical keyboard:

```
hyprctl switchxkblayout at-translated-set-2-keyboard next
```

Note

If you want a single variant ie. pl/dvorak on one layout but us/qwerty on the other, xkb parameters can still be blank, however the amount of comma-separated parameters have to match. Alternatively, a single parameter can be specified for it to apply to all three.

```
input {  
    kb_layout = pl,us,ru  
    kb_variant = dvorak,,  
    kb_options = caps:ctrl_modifier  
}
```

seterror

Sets the hyprctl error string. Will reset when Hyprland's config is reloaded.

```
hyprctl seterror 'rgba(66ee66ff)' hello world this is my problem
```

or disable:

```
hyprctl seterror disable
```

setprop

Sets a window prop. Can be locked by adding `lock` at the end. If `lock` is not added, will be unlocked. Locking means a dynamic windowrule *cannot* override this setting.

Prop List:

| prop | comment |
|----------------|--------------------------|
| animationstyle | string, cannot be locked |

| prop | comment |
|-----------------------|---|
| rounding | int, -1 means not overridden |
| bordersize | int, -1 means not overridden |
| forcenoblur | 0/1 |
| forceopaque | 0/1 |
| forceopaqueoverridden | 0/1 |
| forceallowsinput | 0/1, forceinput rule |
| forcenoanim | 0/1 |
| forcenoborder | 0/1 |
| forcenodim | 0/1 |
| forcenoshadow | 0/1 |
| windowdancecompat | 0/1 |
| nomaxsize | 0/1 |
| dimarounds | 0/1 |
| keepaspectratio | 0/1 |
| alphaoverride | 0/1, makes the next setting be override instead of multiply |
| alpha | float 0.0 - 1.0 |
| alphainactiveoverride | 0/1, makes the next setting be override instead of multiply |
| alphainactive | float 0.0 - 1.0 |
| activebordercolor | color, -1 means not set |
| inactivebordercolor | color, -1 means not set |

```
hyprctl setprop address:0x13371337 forcenoanim 1 lock # with locking
```

```
hyprctl setprop address:0x13371337 nomaxsize 0           # without locking
```

notify

Sends a notification using the built-in Hyprland notification system.

```
hyprctl notify [ICON] [TIME_MS] [COLOR] [MESSAGE]
```

For example:

```
hyprctl notify -1 10000 "rgb(ff1ea3)" "Hello everyone!"
```

Icon of -1 means “No icon”

Color of 0 means “Default color for icon”

Icon list:

```
WARNING = 0
INFO = 1
HINT = 2
ERROR = 3
CONFUSED = 4
OK = 5
```

Info

```
version - prints the hyprland version, meaning flags, commit and branch of build.
monitors - lists active outputs with their properties, 'monitors all' lists active and inactive outputs
workspaces - lists all workspaces with their properties
activeworkspace - gets the active workspace and its properties
workspacerules - gets the list of defined workspace rules
clients - lists all windows with their properties
devices - lists all connected keyboards and mice
decorations [window]- lists all decorations and their info
binds - lists all registered binds
activewindow - gets the active window name and its properties
layers - lists all the layers
splash - prints the current random splash
getoption [option] - gets the config option status (values)
cursorpos - gets the current cursor pos in global layout coordinates
```

```
animations - gets the current config'd info about animations and beziers  
instances - lists all running instances of hyprland with their info  
layouts - lists all layouts available (including plugin'd ones)
```

For the getoption command, the option name should be written as `section:option` , e.g.:

```
hyprctl getoption general:border_size  
# For nested sections:  
hyprctl getoption input:touchpad:disable_while_ttyping
```

See Variables for section and options you can use.

Batch

You can also use `--batch` to specify a batch of commands to execute

e.g.

```
hyprctl --batch "keyword general:border_size 2 ; keyword general:gaps_out 20"
```

`;` separates the commands

Flags

You can specify flags for the request like this:

```
hyprctl -j monitors
```

flag list:

```
j -> output in JSON  
i -> select instance (id or index in hyprctl instances)
```



[/Plugins/Using Plugins](#)

Using Plugins

This page will tell you how to use plugins.

- Disclaimers
- Getting plugins
- Installing / Using plugins
 - hyprpm
 - Manual
- FAQ About Plugins
 - My Hyprland crashes!
 - How do I list my loaded plugins?
 - How do I make my own plugin?
 - Where do I find plugins?
 - Are plugins safe?
 - Do plugins decrease Hyprland's stability?

Disclaimers

Warning

Plugins are written in C++ and will run as a part of Hyprland.

Make sure to *always* read the source code of the plugins you are going to use and to trust the source.

Writing a plugin to wipe your computer is easy.

Never trust random .so files you receive from other people.

Getting plugins

Plugins come as *shared objects*, aka. `.so` files.

Hyprland does not have any “default” plugins, so any plugin you may want to use you will have to find yourself.

Installing / Using plugins

It is *highly* recommended you use the Hyprland Plugin Manager, `hyprpm`. For manual instructions, see a bit below.

hyprpm

Make sure you have the required dependencies: `cpio`, `meson`, `cmake`.

Find a repository you want to install plugins from. As an example, we will use `hyprland-plugins`.

```
hyprpm add https://github.com/hyprwm/hyprland-plugins
```

once it finishes, you can list your installed plugins with

```
hyprpm list
```

and enable or disable them via `hyprpm enable name` and `hyprpm disable name`.

In order for the plugins to be loaded into hyprland, run `hyprpm reload`.

You can add `exec-once = hyprpm reload -n` to your hyprland config to have plugins loaded at startup. `-n` will make hyprpm send a notification if anything goes wrong (e.g. update needed)

In order update your plugins, run `hyprpm update`.

For all options of `hyprpm`, run `hyprpm -h`.

Manual

Different plugins may have different build methods, refer to their instructions.

If you don’t have hyprland headers installed, clone hyprland, checkout to your version, build hyprland, and run `sudo make installheaders`. Then build your plugin(s).

To load plugins manually, use `hyprctl plugin load path` !NOTE: Path HAS TO BE ABSOLUTE!

You can unload plugins with `hyprctl plugin unload path`.

FAQ About Plugins

My Hyprland crashes!

Oh no. Oopsie. Usually means a plugin is broken. `hyprpm disable` it.

How do I list my loaded plugins?

```
hyprctl plugin list
```

How do I make my own plugin?

See here.

Where do I find plugins?

Try looking around here. You can also see a list at awesome-hyprland. Note it may not be complete.

Are plugins safe?

As long as you read the source code of your plugin(s) and can see there's nothing bad going on, they will be safe.

Do plugins decrease Hyprland's stability?

Hyprland employs a few tactics to unload plugins that crash. However, those tactics may not always work. In general, as long as the plugin is well-designed, it should not affect the stability of Hyprland.



 /Configuring/Variables

Variables

For basic syntax info, see Configuring Hyprland.

This page documents all the “options” of Hyprland. For binds, monitors, animations, etc. see the sidebar. For anything else, see Keywords.

Please keep in mind some options that are layout-specific will be documented in the layout pages and not here. (See the Sidebar for Dwindle and Master layouts)

- Variable types
- Sections
 - General
 - Decoration
 - Blur
 - Animations
 - Input
 - Touchpad
 - Touchdevice
 - Tablet
 - Per-device input config
 - Gestures
 - Group
 - Groupbar
 - Misc
 - Binds
 - XWayland
 - OpenGL
 - Debug
 - More

Variable types

| type | description |
|----------|--|
| int | integer |
| bool | boolean, true or false (yes or no, on or off, 0 or 1) - any numerical value that is not 0 or 1 will cause undefined behavior. |
| float | floating point number |
| color | color (see hint below for color info) |
| vec2 | vector with 2 values (float), separated by a space (e.g. 0 0 or -10.9 99.1) |
| MOD | a string modmask (e.g. SUPER or SUPERSHIFT or SUPER + SHIFT or SUPER and SHIFT or CTRL_SHIFT or empty for none. You are allowed to put any separators you please except for a ,) |
| str | a string |
| gradient | a gradient, in the form of color color ... [angle] where color is a color (see above) and angle is an angle in degrees, in the format of 123deg e.g. 45deg (e.g. rgba(11ee11ff) rgba(1111eff) 45deg) Angle is optional and will default to 0deg |

Info

Colors:

You have 3 options:

rgba(), e.g. rgba(b3ff1aee)

rgb(), e.g. rgb(b3ff1a)

legacy, e.g. 0xeb3ff1a -> ARGB order

Mod list:

SHIFT CAPS CTRL/CONTROL ALT MOD2 MOD3 SUPER/WIN/LOGO/MOD4 MOD5

Sections

General

| name | description | type | default |
|-----------------------|--|----------|-------------|
| sensitivity | mouse sensitivity (legacy, may cause bugs if not 1, prefer <code>input:sensitivity</code>) | float | 1.0 |
| border_size | size of the border around windows | int | 1 |
| no_border_on_floating | disable borders for floating windows | bool | false |
| gaps_in | gaps between windows | int | 5 |
| gaps_out | gaps between windows and monitor edges | int | 20 |
| gaps_workspaces | gaps between workspaces. Stacks with gaps_out. | int | 0 |
| col.inactive_border | border color for inactive windows | gradient | 0xffffffff |
| col.active_border | border color for the active window | gradient | 0xff444444 |
| col.nogroup_border | inactive border color for window that cannot be added to a group (see <code>denywindowfromgroup</code> <code>dispatcher</code>) | gradient | 0xfffffaaff |

| name | description | type | default |
|---------------------------|--|-------------|----------------|
| col.nogroup_border_active | active border color for window that cannot be added to a group | gradient | 0xffff00ff |
| cursor_inactive_timeout | in seconds, after how many seconds of cursor's inactivity to hide it. Set to 0 for never. | int | 0 |
| layout | which layout to use. [dwindle/master] | str | dwindle |
| no_cursor_warp | if true, will not warp the cursor in many cases (focusing, keybinds, etc) | bool | false |
| no_focusFallback | if true, will not fall back to the next available window when moving focus in a direction where no window was found | bool | false |
| apply_sens_to_raw | if on, will also apply the sensitivity to raw mouse output (e.g. sensitivity in games) NOTICE: <i>really</i> not recommended. | bool | false |
| resize_on_border | enables resizing windows by clicking and dragging on borders and gaps | bool | false |
| extend_border_grab_area | extends the area around the border where you can click and drag on, only used when general:resize_on_border is on. | int | 15 |
| hover_icon_on_border | show a cursor icon when hovering over borders, only | bool | true |

| name | description | type | default |
|---------------|--|-------------|----------------|
| | used when general:resize_on_border is on. | | |
| allow_tearing | master switch for allowing tearing to occur. See the Tearing page. | bool | false |

Warning

Prefer using `input:sensitivity` over `general:sensitivity` to avoid bugs, especially with Wine/Proton apps.

Decoration

| name | description | type | default |
|---------------------|--|-------------|----------------|
| rounding | rounded corners' radius (in layout px) | int | 0 |
| active_opacity | opacity of active windows. [0.0 - 1.0] | float | 1.0 |
| inactive_opacity | opacity of inactive windows. [0.0 - 1.0] | float | 1.0 |
| fullscreen_opacity | opacity of fullscreen windows. [0.0 - 1.0] | float | 1.0 |
| drop_shadow | enable drop shadows on windows | bool | true |
| shadow_range | Shadow range ("size") in layout px | int | 4 |
| shadow_render_power | in what power to render the falloff (more power, the faster the falloff) [1 - 4] | int | 3 |

| name | description | type | default |
|----------------------|---|-------------|----------------|
| shadow_ignore_window | if true, the shadow will not be rendered behind the window itself, only around it. | bool | true |
| col.shadow | shadow's color. Alpha dictates shadow's opacity. | color | 0xee1a1a1a |
| col.shadow_inactive | inactive shadow color. (if not set, will fall back to col.shadow) | color | unset |
| shadow_offset | shadow's rendering offset. | vec2 | [0, 0] |
| shadow_scale | shadow's scale. [0.0 - 1.0] | float | 1.0 |
| dim_inactive | enables dimming of inactive windows | bool | false |
| dim_strength | how much inactive windows should be dimmed [0.0 - 1.0] | float | 0.5 |
| dim_special | how much to dim the rest of the screen by when a special workspace is open. [0.0 - 1.0] | float | 0.2 |
| dim_around | how much the <code>dimaround</code> window rule should dim by. [0.0 - 1.0] | float | 0.4 |
| screen_shader | a path to a custom shader to be applied at the end of rendering. See <code>examples/screenShader.frag</code> for an example. | str | [[Empty]] |

Blur

Subcategory decoration:blur:

| name | description | type | default |
|-------------------|--|-------------|----------------|
| enabled | enable kawase window background blur | bool | true |
| size | blur size (distance) | int | 8 |
| passes | the amount of passes to perform | int | 1 |
| ignore_opacity | make the blur layer ignore the opacity of the window | bool | false |
| new_optimizations | whether to enable further optimizations to the blur. Recommended to leave on, as it will massively improve performance. | bool | true |
| xray | if enabled, floating windows will ignore tiled windows in their blur. Only available if blur_new_optimizations is true. Will reduce overhead on floating blur significantly. | bool | false |
| noise | how much noise to apply. [0.0 - 1.0] | float | 0.0117 |
| contrast | contrast modulation for blur. [0.0 - 2.0] | float | 0.8916 |
| brightness | brightness modulation for blur. [0.0 - 2.0] | float | 0.8172 |
| vibrancy | Increase saturation of blurred colors. [0.0 - 1.0] | float | 0.1696 |
| vibrancy_darkness | How strong the effect of vibrancy is on dark areas . [0.0 - 1.0] | float | 0.0 |
| special | whether to blur behind the special workspace (note: expensive) | bool | false |
| popups | whether to blur popups (e.g. right-click menus) | bool | false |

| name | description | type | default |
|--------------------|--|-------------|----------------|
| popups_ignorealpha | works like ignorealpha in layer rules. If pixel opacity is below set value, will not blur. [0.0 - 1.0] | float | 0.2 |

Important

A subcategory is a nested category:

```
decoration {
    # ...
    # ...

    blur {
        # ...
        # ...
    }
}
```

Doing `decoration:blur {` is **invalid!**

Info

`blur:size` and `blur:passes` have to be at least 1.

Increasing `blur:passes` is necessary to prevent blur looking wrong on higher `blur:size` values, but remember that higher `blur:passes` will require more strain on the GPU.

Animations

| name | description | type | default |
|------------------------|-------------------------------|-------------|----------------|
| enabled | enable animations | bool | true |
| first_launch_animation | enable first launch animation | bool | true |

Info

More about Animations.

Input

| name | description | type | default |
|--------------------|---|-------------|----------------|
| kb_model | Appropriate XKB keymap parameter. See the note below. | str | [[Empty]] |
| kb_layout | Appropriate XKB keymap parameter | str | us |
| kb_variant | Appropriate XKB keymap parameter | str | [[Empty]] |
| kb_options | Appropriate XKB keymap parameter | str | [[Empty]] |
| kb_rules | Appropriate XKB keymap parameter | str | [[Empty]] |
| kb_file | If you prefer, you can use a path to your custom .xkb file. | str | [[Empty]] |
| numlock_by_default | Engage numlock by default. | bool | false |
| repeat_rate | The repeat rate for held-down keys, in repeats per second. | int | 25 |
| repeat_delay | Delay before a held-down key is repeated, in milliseconds. | int | 600 |
| sensitivity | Sets the mouse input sensitivity. Value will be clamped to the range -1.0 to 1.0. libinput#pointer-acceleration | float | 0.0 |
| accel_profile | Sets the cursor acceleration profile. Can be one of adaptive , | str | [[Empty]] |

| name | description | type | default |
|--------------------|--|-------------|----------------|
| | flat . Can also be custom , see below. Leave empty to use libinput 's default mode for your input device. libinput#pointer-acceleration [adaptive/flat/custom] | | |
| force_no_accel | Force no cursor acceleration. This bypasses most of your pointer settings to get as raw of a signal as possible. Enabling this is not recommended due to potential cursor desynchronization. | bool | false |
| left_handed | Switches RMB and LMB | bool | false |
| scroll_points | Sets the scroll acceleration profile, when accel_profile is set to custom . Has to be in the form <step> <points> . Leave empty to have a flat scroll curve. | str | [[Empty]] |
| scroll_method | Sets the scroll method. Can be one of 2fg (2 fingers), edge , on_button_down , no_scroll . libinput#scrolling [2fg/edge /on_button_down/no_scroll] | str | [[Empty]] |
| scroll_button | Sets the scroll button. Has to be an int, cannot be a string. Check wev if you have any doubts regarding the ID. 0 means default. | int | 0 |
| scroll_button_lock | If the scroll button lock is enabled, the button does not | bool | 0 |

| name | description | type | default |
|-----------------------------|---|-------------|----------------|
| | need to be held down. Pressing and releasing the button once enables the button lock, the button is now considered logically held down. Pressing and releasing the button a second time logically releases the button. While the button is logically held down, motion events are converted to scroll events. | | |
| natural_scroll | Inverts scrolling direction. When enabled, scrolling moves content directly instead of manipulating a scrollbar. | bool | false |
| follow_mouse | Specify if and how cursor movement should affect window focus. See the note below. [0/1/2/3] | int | 1 |
| mouse_refocus | If disabled and <code>follow_mouse=1</code> then mouse focus will not switch to the hovered window unless the mouse crosses a window boundary. | bool | true |
| float_switch_override_focus | If enabled (1 or 2), focus will change to the window under the cursor when changing from tiled-to-floating and vice versa. If 2, focus will also follow mouse on float-to-float switches. | int | 1 |
| special_fallthrough | if enabled, having only floating windows in the special workspace will not block focusing windows in | bool | false |

| name | description | type | default |
|-------------|------------------------|-------------|----------------|
| | the regular workspace. | | |

Info

XKB Settings

You can find a list of models, layouts, variants and options in `/usr/share/X11/xkb/rules/base.1st`. Alternatively, you can use the `localectl` command to discover what is available on your system.

For switchable keyboard configurations, take a look at the [uncommon tips & tricks](#) page entry.

Info

Follow Mouse Cursor

- 0 - Cursor movement will not change focus.
- 1 - Cursor movement will always change focus to the window under the cursor.
- 2 - Cursor focus will be detached from keyboard focus. Clicking on a window will move keyboard focus to that window.
- 3 - Cursor focus will be completely separate from keyboard focus. Clicking on a window will not change keyboard focus.

Custom accel profiles

`accel_profile`

`custom <step> <points...>`

for example `custom 200 0.0 0.5`

`scroll_points`

NOTE: Only works when `accel_profile` is set to `custom`.

`<step> <points...>`

For example `0.2 0.0 0.5 1 1.2 1.5`

To mimic the Windows acceleration curves, take a look at this script.

See the libinput doc for more insights on how it works.

Touchpad

Subcategory `input:touchpad`:

| name | description | type | default |
|-------------------------|---|-------------|----------------|
| disable_while_typing | Disable the touchpad while typing. | bool | true |
| natural_scroll | Inverts scrolling direction. When enabled, scrolling moves content directly instead of manipulating a scrollbar. | bool | false |
| scroll_factor | Multiplier applied to the amount of scroll movement. | float | 1.0 |
| middle_button_emulation | Sending LMB and RMB simultaneously will be interpreted as a middle click. This disables any touchpad area that would normally send a middle click based on location. libinput#middle-button-emulation | bool | false |
| tap_button_map | Sets the tap button mapping for touchpad button emulation. Can be one of <code>lrm</code> (default) or <code>lmr</code> (Left, Middle, Right Buttons). [lrm/lmr] | str | [[Empty]] |
| clickfinger_behavior | Button presses with 1, 2, or 3 fingers will be mapped to LMB, RMB, and MMB respectively. This disables interpretation of clicks based on location on the touchpad. libinput#clickfinger-behavior | bool | false |

| name | description | type | default |
|--------------|---|-------------|----------------|
| tap-to-click | Tapping on the touchpad with 1, 2, or 3 fingers will send LMB, RMB, and MMB respectively. | bool | true |
| drag_lock | When enabled, lifting the finger off for a short time while dragging will not drop the dragged item. libinput#tap-and-drag | bool | false |
| tap-and-drag | Sets the tap and drag mode for the touchpad | bool | false |

Touchdevice

Subcategory *input:touchdevice*:

| name | description | type | default |
|-------------|---|-------------|----------------|
| transform | transform the input from touchdevices. The possible transformations are the same as those of the monitors | int | 0 |
| output | the monitor to bind touch devices. Empty means unset and will use the current / autodetected. | string | [[Empty]] |
| enabled | Whether input is enabled for touch devices. | bool | true |

Tablet

Subcategory *input:tablet*:

| name | description | type | default |
|-------------|---|-------------|----------------|
| transform | transform the input from tablets. The possible transformations are the same as those of the | int | 0 |

| name | description | type | default |
|-----------------|--|-------------|----------------|
| | monitors | | |
| output | the monitor to bind tablets. Empty means unset and will use the current / autodetected. | string | [[Empty]] |
| region_position | position of the mapped region in monitor layout. | vec2 | [0, 0] |
| region_size | size of the mapped region. When this variable is set, tablet input will be mapped to the region. [0, 0] or invalid size means unset. | vec2 | [0, 0] |
| relative_input | whether the input should be relative | bool | false |

Per-device input config

Described here.

Gestures

| name | description | type |
|------------------------------------|--|-------------|
| workspace_swipe | enable workspace swipe gesture | bool |
| workspace_swipe_fingers | how many fingers for the gesture | int |
| workspace_swipe_distance | in px, the distance of the gesture | int |
| workspace_swipe_invert | invert the direction | bool |
| workspace_swipe_min_speed_to_force | minimum speed in px per timepoint to force the change ignoring cancel_ratio . Setting to 0 will disable this | int |

| name | description | type |
|--|--|-------------|
| | mechanic. | |
| workspace_swipe_cancel_ratio | how much the swipe has to proceed in order to commence it. (0.7 -> if > 0.7 * distance, switch, if less, revert) [0.0 - 1.0] | float |
| workspace_swipe_create_new | whether a swipe right on the last workspace should create a new one. | bool |
| workspace_swipe_direction_lock | if enabled, switching direction will be locked when you swipe past the <code>direction_lock_threshold</code> . | bool |
| workspace_swipe_direction_lock_threshold | in px, the distance to swipe before direction lock activates. | int |
| workspace_swipe_forever | if enabled, swiping will not clamp at the neighboring workspaces but continue to the further ones. | bool |
| workspace_swipe_numbered | if enabled, swiping will swipe on consecutive numbered workspaces. | bool |
| workspace_swipe_use_r | if enabled, swiping will use the <code>r</code> prefix instead of the <code>m</code> prefix for finding workspaces. (requires disabled <code>workspace_swipe_numbered</code>) | bool |

Group

| name | description | type | default |
|----------------------------|---|-------------|----------------|
| insert_after_current | whether new windows in a group spawn after current or at group tail | bool | true |
| focus_removed_window | whether Hyprland should focus on the window that has just been moved out of the group | bool | true |
| col.border_active | active group border color | gradient | 0x66ffff00 |
| col.border_inactive | inactive (out of focus) group border color | gradient | 0x66777700 |
| col.border_locked_active | active locked group border color | gradient | 0x66ff5500 |
| col.border_locked_inactive | inactive locked group border color | gradient | 0x66775500 |

Groupbar

Subcategory group:groupbar:

| name | description | type | default |
|-------------|--------------------------------------|-------------|----------------|
| enabled | enables groupbars | bool | true |
| font_family | font used to display groupbar titles | string | Sans |
| font_size | font size for the above | int | 8 |
| gradients | enables gradients | bool | true |

| name | description | type | default |
|---------------------|---|-------------|----------------|
| height | height of the groupbar | int | 14 |
| priority | sets the decoration priority for groupbars | int | 3 |
| render_titles | whether to render titles in the group bar decoration | bool | true |
| scrolling | whether scrolling in the groupbar changes group active window | bool | true |
| text_color | controls the group bar text color | color | 0xffffffff |
| col.active | active group border color | gradient | 0x66ffff00 |
| col.inactive | inactive (out of focus) group border color | gradient | 0x66777700 |
| col.locked_active | active locked group border color | gradient | 0x66ff5500 |
| col.locked_inactive | inactive locked group border color | gradient | 0x66775500 |

Misc

| name | description | type | default |
|--------------------------|---|-------------|----------------|
| disable_hyprland_logo | disables the random hyprland logo / anime girl background. :(| bool | false |
| disable_splash_rendering | disables the hyprland splash rendering. (requires a monitor reload to take effect) | bool | false |
| force_default_wallpaper | Enforce any of the 3 default wallpapers. | int | -1 |

| name | description | type | default |
|---------------------------|--|-------------|----------------|
| | Setting this to 0 disables the anime background. -1 means “random” [-1 - 3] | | |
| vfr | controls the VFR status of hyprland. Heavily recommended to leave on true to conserve resources. | bool | true |
| vrr | controls the VRR (Adaptive Sync) of your monitors. 0 - off, 1 - on, 2 - fullscreen only [0/1/2] | int | 0 |
| mouse_move_enables_dpms | If DPMS is set to off, wake up the monitors if the mouse moves. | bool | false |
| key_press_enables_dpms | If DPMS is set to off, wake up the monitors if a key is pressed. | bool | false |
| always_follow_on_dnd | Will make mouse focus follow the mouse when drag and dropping. Recommended to leave it enabled, especially for people using focus follows mouse at 0. | bool | true |
| layers_hog_keyboard_focus | If true, will make keyboard-interactive layers keep their focus on mouse move (e.g. wofi, bemenu) | bool | true |

| name | description | type | default |
|------------------------------|--|-------------|----------------|
| animate_manual_resizes | If true, will animate manual window resizes/moves | bool | false |
| animate_mouse_windowdragging | If true, will animate windows being dragged by mouse, note that this can cause weird behavior on some curves | bool | false |
| disable_autoreload | If true, the config will not reload automatically on save, and instead needs to be reloaded with <code>hyprctl reload</code> . Might save on battery. | bool | false |
| enable_swallow | Enable window swallowing | bool | false |
| swallow_regex | The <i>class</i> regex to be used for windows that should be swallowed (usually, a terminal). To know more about the list of regex which can be used use this cheatsheet. | str | [[Empty]] |
| swallow_exception_regex | The <i>title</i> regex to be used for windows that should <i>not</i> be swallowed by the windows specified in <code>swallow_regex</code> (e.g. wev). The regex is matched against the parent (e.g. Kitty) window's title on the assumption that it | str | [[Empty]] |

| name | description | type | default |
|----------------------------|---|-------------|----------------|
| | changes to whatever process it's running. | | |
| focus_on_activate | Whether Hyprland should focus an app that requests to be focused (an <code>activate</code> request) | bool | false |
| no_direct_scanout | Disables direct scanout. Direct scanout attempts to reduce lag when there is only one fullscreen application on a screen (e.g. game). It is also recommended to set this to true if the fullscreen application shows graphical glitches. | bool | true |
| hide_cursor_on_touch | Hides the cursor when the last input was a touch input until a mouse input is done. | bool | true |
| mouse_move_focuses_monitor | Whether mouse moving into a different monitor should focus it | bool | true |
| suppress_portal_warnings | disables warnings about incompatible portal implementations. | bool | false |
| render_ahead_of_time | [Warning: buggy] starts rendering <i>before</i> your monitor displays a frame in order to lower latency | bool | false |

| name | description | type | default |
|---------------------------------|---|-------------|----------------|
| render_ahead_safezone | how many ms of safezone to add to rendering ahead of time. Recommended 1-2. | int | 1 |
| cursor_zoom_factor | the factor to zoom by around the cursor. AKA. Magnifying glass. Minimum 1.0 (meaning no zoom) | float | 1.0 |
| cursor_zoom_rigid | whether the zoom should follow the cursor rigidly (cursor is always centered if it can be) or loosely | bool | false |
| allow_session_lock_restore | if true, will allow you to restart a lockscreen app in case it crashes (red screen of death) | bool | false |
| background_color | change the background color. (requires enabled disable_hyprland_logo) | color | 0x111111 |
| close_special_on_empty | close the special workspace if the last window is removed | bool | true |
| new_window_takes_overFullscreen | if there is a fullscreen window, whether a new tiled window opened should replace the fullscreen one or stay behind. 0 - behind, 1 - takes over, 2 - unfullscreen the current | int | 0 |

| name | description | type | default |
|-------------|---------------------------|-------------|----------------|
| | fullscreen window [0/1/2] | | |

Binds

| name | description | type | default |
|--------------------------|---|-------------|----------------|
| pass_mouse_when_bound | if disabled, will not pass the mouse events to apps / dragging windows around if a keybind has been triggered. | bool | false |
| scroll_event_delay | in ms, how many ms to wait after a scroll event to allow to pass another one for the binds. | int | 300 |
| workspace_back_and_forth | If enabled, an attempt to switch to the currently focused workspace will instead switch to the previous workspace. Akin to i3's <i>auto_back_and_forth</i> . | bool | false |
| allow_workspace_cycles | If enabled, workspaces don't forget their previous workspace, so cycles can be created by switching to the first workspace in a sequence, then endlessly going to the previous workspace. | bool | false |
| workspace_center_on | Whether switching workspaces should center the cursor on the workspace (0) or on the last active window for that workspace (1) | int | 0 |
| focus_preferred_method | sets the preferred focus finding | int | 0 |

| name | description | type | default |
|----------------------------|---|-------------|----------------|
| | method when using <code>focuswindow / movewindow /etc</code> with a direction. 0 - history (recent have priority), 1 - length (longer shared edges have priority) | | |
| ignore_group_lock | If enabled, dispatchers like <code>moveintogroup</code> , <code>moveoutofgroup</code> and <code>movewindowingroup</code> will ignore lock per group. | bool | false |
| movefocus_cyclesFullscreen | If enabled, when on a fullscreen window, <code>movefocus</code> will cycle fullscreen, if not, it will move the focus in a direction. | bool | true |

XWayland

| name | description | type | default |
|----------------------|---|-------------|----------------|
| use_nearest_neighbor | uses the nearest neighbor filtering for xwayland apps, making them pixelated rather than blurry | bool | true |
| force_zero_scaling | forces a scale of 1 on xwayland windows on scaled displays. | bool | false |

OpenGL

| name | description | type | default |
|---------------------|---|-------------|----------------|
| nvidia_anti_flicker | reduces flickering on nvidia at the cost of | bool | true |

| name | description | type | default |
|-------------|---|-------------|----------------|
| | possible frame drops on lower-end GPUs. On non-nvidia, this is ignored. | | |

Debug

Warning

Only for developers.

| name | description | type | default |
|--------------------|---|-------------|----------------|
| overlay | print the debug performance overlay. Disable VFR for accurate results. | bool | false |
| damage_blink | (epilepsy warning!) flash areas updated with damage tracking | bool | false |
| disable_logs | disable logging to a file | bool | true |
| disable_time | disables time logging | bool | true |
| damage_tracking | redraw only the needed bits of the display. Do not change. (default: full - 2) monitor - 1, none - 0 | int | 2 |
| enable_stdout_logs | enables logging to stdout | bool | false |
| manual_crash | set to 1 and then back to 0 to crash Hyprland. | int | 0 |
| suppress_errors | if true, do not display config file parsing errors. | bool | false |
| watchdog_timeout | sets the timeout in seconds for watchdog to abort processing of a signal of the main thread. Set to 0 to disable. | int | 5 |

| name | description | type | default |
|----------------------|---|-------------|----------------|
| disable_scale_checks | disables verifying of the scale factors. Will result in pixel alignment and rounding errors. | bool | false |

More

There are more config options described in other pages, which are layout- or circumstance-specific. See the sidebar for more pages.

The Hyprland Wiki, built with Hugo.

Hyprland Wiki

 /Useful Utilities/Wallpapers

Wallpapers

Launch your wallpaper utility with `exec-once=`.

hyrpaper

Straight from our oven, `hyrpaper` is a wallpaper utility from the Hypr Development team. See its usage and config by following the README on the GitHub Page.

swaybg

Great utility if all you want is one simple static wallpaper that will never change, and don't want to make a config file. Get it here.

wpaperd

A bit more advanced wallpaper daemon with automatic wallpaper changing options and other fancy stuff. GitHub.

mpvpaper

A neat mpv wrapper to play a video as your wallpaper. GitHub.

swww

An efficient animated wallpaper daemon for wayland, controlled at runtime, which means you can change wallpapers without even needing to restart. GitHub

waypaper

GUI wallpaper manager that allows to visually choose the static or animated wallpapers, and

supports swaybg and swww as backends on wayland. GitHub

The Hyprland Wiki, built with Hugo.



/Configuring/Window Rules

Window Rules

Table of contents

- Table of contents
- Disclaimers
- Window Rules V1
 - Syntax
 - Examples
 - Notes
- Window Rules V2
 - Rules
 - Example Rules
 - Notes
- Layer Rules
 - Rules

Disclaimers

Warning

Window rules (both V1 and V2) are **case sensitive**. (e.g. `firefox` ≠ `Firefox`)

Window Rules V1

You can set window rules to achieve different behaviors from the active container.

Syntax

```
windowrule=RULE,WINDOW
```

- RULE is a rule (and a param if applicable)
- WINDOW is a RegEx, either:
 - plain RegEx (for matching a window class);
 - title: followed by a regex (for matching a window's title)

Examples

```
windowrule=float,^(kitty)$  
windowrule=move 0 0,title:^(Firefox)(.*)$
```

Window Rules V2

In order to allow more flexible rules, while retaining compatibility with the above rule system, window rules V2 were implemented.

In V2, you are allowed to match multiple variables.

the RULE field is unchanged, but in the WINDOW field, you can put regexes for multiple values like so:

```
windowrulev2 = float,class:(kitty),title:(kitty)
```

Tip

In the case of dynamic window titles such as browser windows keep in mind how powerful regex is.

for example a window rule of: `windowrule=opacity 0.3 override 0.3 override,title:(.*)(- Youtube)$` will match any window that contains a string of “-Youtube” after any other text. This could be multiple browser windows or other applications that contain the string for any reason.

for the `windowrulev2 = float,class:(kitty),title:(kitty)` example, the `class:(kitty)` WINDOW field is what keeps the window rule specific to kitty terminals.

For now, the supported fields are:

```
class - class regex
title - title regex
initialClass - initialClass regex
initialTitle - initialTitle regex
xwayland - 0/1
floating - 0/1
fullscreen - 0/1
pinned - 0/1
focus - 0/1
workspace - id or name: and name
onworkspace (how many windows are on the workspace) - int
```

Keep in mind that you *have* to declare at least one field, but not all.

Tip

To get more information about a window's class, title, XWayland status or its size, you can use `hyprctl clients`.

Rules

| Rule | Description | Dynamic |
|---------------------|--|---------|
| float | floats a window | |
| tile | tiles a window | |
| fullscreen | fullscreens a window | |
| fakefullscreen | fakefullscreens a window | |
| maximize | maximizes a window | |
| nofullscreenrequest | prevents windows from requesting fullscreen mode, you can still manually toggle fullscreen. | |
| nomaximizerequest | prevents windows from requesting maximized mode, you can still manually toggle maximize. | |
| move [x] [y] | moves a floating window (x,y -> int or %, e.g. 20% or 100. You are also allowed to do 100%- for the right/bottom anchor, e.g. 100%-20). Additionally, | |

| Rule | Description | Dynamic |
|------------------------------|---|---------|
| | you can also do <code>cursor [x] [y]</code> where x and y are either pixels or percent. Percent is calculated from the window's size. Specify <code>onscreen</code> before other parameters to force the window into the screen (e.g. <code>move onscreen cursor 50% 50%</code>) | |
| <code>size [x] [y]</code> | resizes a floating window (x,y -> int or %, e.g. 20% or 100) | |
| <code>minsize [x] [y]</code> | sets the minimum size on creation (x,y -> int) | |
| <code>maxsize [x] [y]</code> | sets the maximum size on creation (x,y -> int) | |
| <code>center ([opt])</code> | if the window is floating, will center it on the monitor. Set opt as 1 to respect monitor reserved area | |
| <code>pseudo</code> | pseudotiles a window | |
| <code>monitor [id]</code> | sets the monitor on which a window should open. <code>id</code> can be either id or name (either e.g. 1 or e.g. DP-1) | |
| <code>workspace [w]</code> | sets the workspace on which a window should open (for workspace syntax, see dispatchers->workspaces). You can also make [w] to unset , will unset all previous workspace rules applied to this window. You can also add silent after the workspace to make the window open silently. | |
| <code>opacity [a]</code> | additional opacity multiplier. Options for a: <code>float</code> -> sets an opacity OR <code>float float</code> -> sets activeopacity and inactiveopacity respectively. You can also add <code>override</code> after an opacity to make it override instead of a multiplier. (e.g. <code>1.0 override 0.5 override</code>) | ✓ |

| Rule | Description | Dynamic |
|---------------------------|---|---------|
| opaque | forces the window to be opaque (can be toggled with the toggleopaque dispatcher) | ✓ |
| forcergbx | makes hyprland ignore the alpha channel of all the window's surfaces, effectively making it <i>actually, fully 100% opaque</i> | ✓ |
| animation [style] ([opt]) | forces an animation onto a window, with a selected opt. Opt is optional. | ✓ |
| rounding [x] | forces the application to have X pixels of rounding, ignoring the set default (in decoration:rounding). Has to be an int. | ✓ |
| noblur | disables blur for the window | ✓ |
| nofocus | disables focus to the window | |
| noinitfocus | disables the initial focus to the window | |
| noborder | disables borders for the window | ✓ |
| bordersize [size] | sets the border size | ✓ |
| nodim | disables window dimming for the window | ✓ |
| noshadow | disables shadows for the window | ✓ |
| forceinput | forces an XWayland window to receive input, even if it requests not to do so. (Might fix issues like e.g. Game Launchers not receiving focus for some reason) | |
| windowdance | forces an XWayland window to never refocus, used for games/applications like Rhythm Doctor | |
| pin | pins the window (i.e. show it on all workspaces) <i>note: floating only</i> | |

| Rule | Description | Dynamic |
|--------------------|---|---------|
| noanim | disables the animations for the window | ✓ |
| keepaspectratio | forces aspect ratio when resizing window with the mouse | ✓ |
| bordercolor [c] | force the bordercolor of the window. Options for c: color / color ... color angle -> sets the active border color/gradient OR color color / color ... color angle color ... color [angle] -> sets the active and inactive border color/gradient of the window. See variables->colors for color definition. | ✓ |
| idleinhibit [mode] | sets an idle inhibit rule for the window. If active, apps like swayidle will not fire. Modes: none , always , focus , fullscreen | ✓ |
| unset | removes all previously set rules for the given parameters. Please note it has to match EXACTLY. | |
| nomaxsize | removes max size limitations. Especially useful with windows that report invalid max sizes (e.g. winecfg) | |
| dimaround | dims everything around the window . Please note this rule is meant for floating windows and using it on tiled ones may result in strange behavior. | ✓ |
| stayfocused | forces focus on the window as long as it's visible | |
| xray [on] | sets blur xray mode for the window (0 for off, 1 for on, unset for default) | ✓ |
| group [options] | set window group properties. See the note below. | |
| immediate | forces the window to allow to be torn. See the Tearing page. | ✓ |

| Rule | Description | Dynamic |
|-----------------|--|---------|
| nearestneighbor | forces the window to use the nearest neighbor filtering. | ✓ |

Info

group **window rule options**

- `set [always]` - Open window as a group.
- `new` - Shorthand of `barred set`.
- `lock [always]` - Lock the group that added this window. Use with `set` or `new` (i.e. `new lock`) to create a new locked group.
- `barred` - Do not add the window to the focused group. By default, a window with a `group set` rule will be added to an active group if possible.
- `deny` - Do not allow window to be toggled as or added to group (see `denywindowfromgroup` dispatcher).
- `invade` - Force open window in the locked group.
- `override [other options]` - Override other `group` rules, e.g. You can make all windows in a particular workspace open as a group, and use `group override barred` to make windows with specific titles open as normal windows.
- `unset` - Clear all `group` rules.

The `group` rule without options is a shorthand for `group set`.

By default, `set` and `lock` only affect new windows once. The `always` qualifier makes them always effective.

Example Rules

```
windowrule = move 100 100,^(kitty)$ # moves kitty to 100 100
windowrule = animation popin,^(kitty)$ # sets the animation style for kitty
windowrule = noblur,^(firefox)$ # disables blur for firefox
windowrule = move cursor -50% -50%,^(kitty)$ # moves kitty to the center of
the cursor
windowrulev2 = bordercolor rgb(FF0000) rgb(880808),fullscreen:1 # set
bordercolor to red if window is fullscreen
windowrulev2 = bordercolor rgb(FFFF00),title:^(.*Hyprland.*)$ # set
bordercolor to yellow when title contains Hyprland
windowrule = opacity 1.0 override 0.5 override,^(kitty)$ # set opacity to 1.0
```

```
active and 0.5 inactive for kitty  
windowrule = rounding 10,^(kitty)$ # set rounding to 10 for kitty
```

Notes

Rules that are marked as *Dynamic* will be reevaluated if the matching property of the window changes. For instance, if a rule is defined that changes the bordercolor of a window when it is floating, then the bordercolor will change to the requested color when it is set to floating, and revert to the default color when it is tiled again.

Rules will be processed from top to bottom, where the *last* match will take precedence. i.e.

```
windowrulev2 = opacity 0.8 0.8, class:^(kitty)$  
windowrulev2 = opacity 0.5 0.5, floating:1
```

-> all kitty windows will have opacity 0.8, except if they are floating. Then they will have opacity 0.5. -> all floating windows will have opacity 0.5.

```
windowrulev2 = opacity 0.5 0.5, floating:1  
windowrulev2 = opacity 0.8 0.8, class:^(kitty)$
```

-> all kitty windows will have opacity 0.8, also if they are floating. -> all other floating windows will have opacity 0.5.

Tip

Opacity is *always* a PRODUCT of all opacities. E.g. active_opacity to 0.5 and windowrule opacity to 0.5 will result in a total opacity 0.25. You are allowed to set opacities over 1, but any opacity product over 1 will cause graphical glitches. E.g. 0.5 * 2 = 1 , and it will be fine, 0.5 * 4 will cause graphical glitches.

Layer Rules

Some things in wayland are not windows, but layers. That includes for example most launchers, your status bar or wallpaper.

Those have specific rules separate from windows:

```
layerrule = rule, namespace  
# or
```

```
layerrule = rule, address
```

where `rule` is the rule and `namespace` is the namespace regex (find namespaces in `hyprctl layers`) or `address` is an address in the form of `address:0x[hex]`

Rules

| rule | description |
|-----------------|---|
| unset | removes all layerRules previously set for a select namespace regex. Please note it has to match <i>exactly</i> |
| noanim | disables animations |
| blur | enables blur for the layer |
| ignorealpha [a] | makes blur ignore pixels with opacity of <code>a</code> or lower. <code>a</code> is float value from 0 to 1. <code>a = 0</code> if unspecified. |
| ignorezero | makes blur ignore fully transparent pixels. Same as <code>ignorealpha 0</code> . |
| xray [on] | sets the blur xray mode for a layer. 0 for off, 1 for on, unset for default. |

The Hyprland Wiki, built with Hugo.



 /Configuring/Workspace Rules

Workspace Rules

Table of contents

- Table of contents
- Workspace Rules
 - Syntax
 - Examples
 - Rules
 - Example Rules

Workspace Rules

You can set workspace rules to achieve workspace-specific behaviors. For instance, you can define a workspace where all windows are drawn without borders or gaps.

For layout-specific rules, see the specific layout page. For example: Master Layout->Workspace Rules

Syntax

```
workspace=WORKSPACE, RULES
```

- WORKSPACE is a valid workspace identifier (see Dispatchers->Workspaces). This field is mandatory;
- RULES is one (or more) rule(s) as described here in rules.

Examples

```
workspace=name:myworkspace, gapsin:0, gapsout:0
workspace=3, rounding:false, bordersize:0
```

Rules

| Rule | Description | type |
|----------------------|--|--------|
| monitor:[m] | Binds a workspace to a monitor See syntax and Monitors. | string |
| default:[b] | Whether this workspace should be the default workspace for the given monitor | bool |
| gapsin:[x] | Set the gaps between windows (equivalent to General->gaps_in) | int |
| gapsout:[x] | Set the gaps between windows and monitor edges (equivalent to General->gaps_out) | int |
| bordersize:[x] | Set the border size around windows (equivalent to General->border_size) | int |
| border:[b] | Whether to draw borders or not | bool |
| shadow:[b] | Whether to draw shadows or not | bool |
| rounding:[b] | Whether to draw rounded windows or not | bool |
| decorate:[b] | Whether to draw window decorations or not | bool |
| persistent:[b] | Keep this workspace alive even if empty and inactive | bool |
| on-created-empty:[c] | A command to be executed once a workspace is created empty (i.e. not created by moving a window to it). See the command syntax | string |

Example Rules

```
workspace = 3, rounding:false, decorate:false
workspace = name:coding, rounding:false, decorate:false, gapsin:0, gapsout:0,
```

```
border:false, decorate:false, monitor:DP-1
workspace = 8,bordersize:8
workspace = name:Hello, monitor:DP-1, default:true
workspace = name:gaming, monitor:desc:Chimei Innolux Corporation 0x150C,
default:true
workspace = 5, on-created-empty:[float] firefox
workspace = special:scratchpad, on-created-empty:foot
```

The Hyprland Wiki, built with Hugo.

Hyprland Wiki

/Configuring/Xwayland

Xwayland

XWayland is the bridging mechanism between legacy Xorg programs and Wayland compositors.

HiDPI XWayland

XWayland currently looks pixelated on HiDPI screens, due to Xorg's inability to scale.

This problem is mitigated by the `xwayland:force_zero_scaling` option, which forces XWayland windows not to be scaled.

This will get rid of the pixelated look, but will not scale applications properly. To do this, each toolkit has its own mechanism.

```
# change monitor to high resolution, the last argument is the scale factor
monitor=,highres,auto,2

# unscale XWayland
xwayland {
    force_zero_scaling = true
}

# toolkit-specific scale
env = GDK_SCALE,2
env = XCURSOR_SIZE,32
```

The `GDK_SCALE` variable won't conflict with Wayland-native GTK programs.

Important

XWayland HiDPI patches are no longer supported. Do not use them.

The Hyprland Wiki, built with Hugo.

Anyrun A wayland native krunner-like runner, made with customizability in mind. # Features - Style customizability with GTK+ CSS - More info in [Styling](#Styling) - Can do basically anything - As long as it can work with input and selection - Hence the name anyrun - Easy to make plugins - You only need 4 functions! - See [Rink](plugins/rink) for a simple example. More info in the documentation of the anyrun-plugin crate. - Responsive - Asynchronous running of plugin functions - Wayland native - GTK layer shell for overlaying the window - data-control for managing the clipboard

Usage ## Dependencies Anyrun mainly depends various GTK libraries, and rust of course for building the project. Rust you can get with [rustup](<https://rustup.rs>). The rest are statically linked in the binary. Here are the libraries you need to have to build & run it: - `gtk-layer-shell (libgtk-layer-shell)` - `gtk3 (libgtk-3 libgdk-3)` - `pango (libpango-1.0)` - `cairo (libcairo libcairo-gobject)` - ` gdk-pixbuf2 (libgdk_pixbuf-2.0)` - `glib2 (libgobject-2.0 libgio-2.0 libglib-2.0)` ## Installation [! [Packaging status](<https://repology.org/badge/vertical-allrepos/anyrun.svg>)](<https://repology.org/project/anyrun/versions>)

Nix You can use the flake: ```nix # flake.nix { inputs = { nixpkgs.url = "github:NixOS/nixpkgs/nixos-unstable"; anyrun.url = "github:Kirottu/anyrun"; anyrun.inputs.nixpkgs.follows = "nixpkgs"; }; outputs = { self, nixpkgs, anyrun }: let in { nixosConfigurations.HOSTNAME = nixpkgs.lib.nixosSystem { # ... system.packages = [anyrun.packages.\${system}.anyrun]; # ... }; }; }``` The flake provides multiple packages: - anyrun (default) - just the anyrun binary - anyrun-with-all-plugins - anyrun and all builtin plugins - applications - the applications plugin - dictionary - the dictionary plugin - kidex - the kidex plugin - randr - the randr plugin - rink - the rink plugin - shell - the shell plugin - stdin - the stdin plugin - symbols - the symbols plugin - translate - the translate plugin - websearch - the websearch plugin

home-manager module We have a home-manager module available at `homeManagerModules.default`. You use it like this: ```nix { programs.anyrun = { enable = true; config = { plugins = [# An array of all the plugins you want, which either can be paths to the .so files, or their packages inputs.anyrun.packages.\${pkgs.system}.applications ./some_plugin.so "\${inputs.anyrun.packages.\${pkgs.system}.anyrun-with-all-plugins}/lib/kidex"]; width = { fraction = 0.3; }; position = "top"; verticalOffset = { absolute = 0; }; hideIcons = false; ignoreExclusiveZones = false; layer = "overlay"; hidePluginInfo = false; closeOnClick = false; showResultsImmediately = false; maxEntries = null; }; extraCss = ".some_class { background: red; }"; extraConfigFiles."some-plugin.ron".text = "Config(// for any other plugin // this file will be put in ~/.config/anyrun/some-plugin.ron // refer to docs of xdg.configFile for available options)"; }; }``` You might also want to use the binary cache to avoid building locally. ```nix nix.settings = { builders-use-substitutes = true; # substituters to use substituters = ["https://anyrun.cachix.org"]; trusted-public-keys = ["anyrun.cachix.org-1:pqBobmOjI7nKlsUMV25u9QHa9btJK65/C8vnO3p346s="]; }``` ## Manual installation Make sure all of the dependencies are installed, and then run the following commands in order: ```sh git clone <https://github.com/Kirottu/anyrun.git> # Clone the repository cd anyrun # Change the active directory to it cargo build --release # Build all the packages cargo install --path anyrun/ # Install the anyrun binary mkdir -p ~/.config/anyrun/plugins # Create the config directory and the plugins subdirectory cp target/release/*.so ~/.config/anyrun/plugins # Copy all of the built plugins to the correct directory cp examples/config.ron ~/.config/anyrun/config.ron # Copy the default config file``` ## Plugins Anyrun requires plugins to function, as they provide the results for input. The list of plugins in this repository is as follows: - [Applications](plugins/applications/README.md) - Search and run system & user specific desktop entries. - [Symbols](plugins/symbols/README.md) - Search unicode symbols. - [Rink](plugins/rink/README.md) - Calculator & unit conversion. - [Shell](plugins/shell/README.md) - Run shell commands. - [Translate](plugins/translate/README.md) - Quickly translate text. - [Kidex](plugins/kidex/README.md) - File search provided by [Kidex](<https://github.com/Kirottu/kidex>). - [Randr](plugins/randr/README.md) - Rotate and resize; quickly change monitor configurations on the fly. - TODO: Only supports Hyprland, needs support for other compositors. - [Stdin](plugins/stdin/README.md) - Turn Anyrun into a dmenu like fuzzy selector. - Should generally be used exclusively with the `--plugins` argument. - [Dictionary](plugins/dictionary/README.md) - Look up definitions for words - [Websearch](plugins/websearch/README.md) - Search the web with configurable engines: Google, Ecosia, Bing, DuckDuckGo.

Configuration The default configuration directory is `~/.config/anyrun` the structure of the config directory is as follows and should be respected by plugins: ``` - anyrun - plugins config.ron style.css``` The [default config file](examples/config.ron) contains the default values, and annotates all configuration options with comments on what they are and how to use them. ## Styling Anyrun supports [GTK+ CSS](<https://docs.gtk.org/gtk3/css-overview.html>) styling. The names for the different widgets and widgets associated with them are as follows: - `entry`: The entry box - `GtkEntry` - `window`: The window - `GtkWindow` - `main`: "Main" parts of the layout - `GtkListBox`: The main list containing the plugins - `GtkBox`: The box combining the main list and the entry box - `plugin`: Anything for the entire plugin - `GtkLabel`: The name of the plugin - `GtkBox`: The different boxes in the plugin view - `GtkImage`: The icon of the plugin - `match`: Widgets of a specific match - `GtkBox`: The main box of the match and the box containing the title and the description if present - `GtkImage`: The icon of the match (if present) - `match-title`: Specific for the title of the match - `GtkLabel` - `match-desc`: Specific for the description of the match - `GtkLabel` ## Arguments The custom arguments for anyrun are as follows: - `--config-dir`, `-c`: Override the configuration directory The rest of the arguments are automatically generated based on the config, and can be used to override configuration parameters. For example if you want to temporarily only run the Applications and Symbols plugins on the top side of the screen, you would run `anyrun --plugins libapplications.so --plugins libsymbols.so --position top`. # Plugin development The plugin API is intentionally very simple

to use. This is all you need for a plugin: `Cargo.toml`:

```
toml #[package] omitted [lib] crate-type = ["cdylib"] # Required to build a dynamic library that can be loaded by anyrun [dependencies] anyrun-plugin = { git = "https://github.com/Kirottu/anyrun" } abi_stable = "0.11.1" # Any other dependencies you may have ``` `lib.rs`:` ``rs use abi_stable::std_types::{RString, RVec, ROption}; use anyrun_plugin::*; #[init] fn init(config_dir: RString) { // Your initialization code. This is run in another thread. // The return type is the data you want to share between functions } #[info] fn info() -> PluginInfo { PluginInfo { name: "Demo".into(), icon: "help-about".into(), // Icon from the icon theme } } # [get_matches] fn get_matches(input: RString) -> RVec { // The logic to get matches from the input text in the `input` argument. // The `data` is a mutable reference to the shared data type later specified. vec![Match { title: "Test match".into(), icon: ROption::RSome("help-about".into()), use_pango: false, description: ROption::RSome("Test match for the plugin API demo".into()), id: ROption::RNone, // The ID can be used for identifying the match later, is not required }].into() } # [handler] fn handler(selection: Match) -> HandleResult { // Handle the selected match and return how anyrun should proceed HandleResult::Close } ``` And that's it! That's all of the API needed to make runners. Refer to the plugins in the [plugins](plugins) folder for more examples.
```

On this page methods

App

This is the main `Gtk.Application` instance that is running.

signals

- `window-toggled` : (`windowName: string, visible: boolean`)
- `config-parsed` : emitted on startup

properties

- `windows` : `Gtk.Window[]`
- `configDir` : `string` path to the config directory

methods

- `addWindow` : (`window: Gtk.Window`) => `void`
- `removeWindow` : (`window: Gtk.Window`) => `void`
- `getWindow` : (`name: string`) => `Gtk.Window`
- `closeWindow` : (`name: string`) => `void`
- `openWindow` : (`name: string`) => `void`
- `toggleWindow` : (`name: string`) => `void`
- `quit` : () => `void`
- `resetCss` : () => `void`

[On this page](#) > methods

```
// this is only signaled for windows exported in config.js
// or added with App.addWindow
const label = Widget.Label()
    .hook(App, (self, windowName, visible) => {
        self.label = `${windowName} is ${visible ? 'visible' : 'not visible'}`;
    }, 'window-toggled')
);
```

Applying CSS

If you want to change the style sheet on runtime

```
// if you apply multiple, they are all going to apply on top of each other
App.applyCss('path-to-file');

// to reset applied stylesheets
App.resetCss();
```

 [Edit page](#)[Previous](#)[Utils](#)[Next](#)[Custom Service](#)

[On this page](#)[Examples](#)

CLI

```
$ ags --help
```

USAGE:

```
ags [OPTIONS]
```

OPTIONS:

| | |
|---------------------|--|
| -h, --help | Print this help and exit |
| -v, --version | Print version and exit |
| -q, --quit | Kill AGS |
| -c, --config | Path to the config file. |
| -b, --bus-name | Bus name of the process |
| -i, --inspector | Open up the Gtk debug tool |
| -t, --toggle-window | Show or hide a window |
| -r, --run-js | Execute string as an async function |
| -f, --run-file | Execute file as an async function |
| -I, --init | Initialize the configuration directory |
| -C, --clear-cache | Remove \$HOME/.cache/ags |

Bus Name

It is possible to run multiple instances with `--bus-name`. It defaults to "ags". When an instance is running, executing `ags` is actually a client process that will try to connect to the instance with the specified bus name.

AGS Wiki



On this page > Examples

```
ags --bus-name test # starts an instance with bus name "test"  
ags -b test -i # opens the inspector on instance with bus name "test"  
  
ags --quit  
ags -b test --quit
```



Tip

The dbus name is `com.github.Aylur.ag.s.<bus-name>` , so the default one is `com.github.Aylur.ag.s.ag.s`

Config file

`--config` defaults to `$HOME/.config/ags/config.js`

```
# example  
ags --config $HOME/.config/some-dir/main.js
```

Toggle Window

`--toggle-window` is just there for the sake of it, if you want to have more control use `--run-js`

```
# example  
ags --toggle-window "bar"
```

AGS Wiki



On this page > Examples

string which will be the body of an *async function* executed relative to `app.ts`.

If there is no `;` character in the string, `return` keyword will be inserted automatically

```
ags -r "'hello'" # prints hello
ags -r "'hello';" # prints undefined
ags -r "return 'hello';" # prints hello
```

`print` will print on the client side, `console.log` and other console methods will log on the main process's `stdout`

```
ags -r "print('hello')" # prints "hello undefined"
ags -r "console.log('hello')" # prints "undefined"
```

`--run-file` reads the content of a file and passes it to `--run-js`

```
# these two are equivalent
ags --run-file /path/to/file.js
ags --run-js "$(cat /path/to/file.js)"
```

It is useful for shebangs

```
#!/usr/bin/env -S ags --run-file
return 'hello from a file'
```

AGS Wiki



On this page > Examples

this throws

```
#!/usr/bin/env -S ags --run-file
import Module from 'file:///path/to/file.js' // throws
```

You can use `import` as an **async** method

```
#!/usr/bin/env -S ags --run-file
const Module = (await import('file:///path/to/file.js')).default;
```

Examples

Let's say you have a `variable` in `$HOME/.config/ags/vars.js` that you want set from cli.

```
vars.js
export const myVar = Variable(0);
```

You can import this module and access this variable

```
ags -r "(await import('file://$HOME/.config/ags/vars.js')).myVar++"
```

You can make `myVar` global, to avoid writing out the import statement

```
vars.js
```

AGS Wiki

[On this page](#)[Examples](#)

```
ags -r "myVar++"
```

 [Edit page](#)[Previous](#) [Type Checking](#)[Next](#) [Widgets](#)

[On this page](#) > [Overview](#)

Config Object

When you start `ags`, it will try to `import` the `default export` from a module which defaults to `~/.config/ags/config.js`. Even if you mutate this object after initialization, the config **will not be reloaded**.

```
config.js
```

```
export default {
    style: './style.css',
    icons: './assests',
    windows: [
        // Array<Gtk.Window>
    ],
    closeWindowDelay: {
        'window-name': 500, // milliseconds
    },
    onConfigParsed: function() {
        // code that runs after this object is loaded
    },
    onWindowToggled: function (windowName, visible) {
        print(`$ {windowName} is ${visible}`)
    },
};
```

The exported config object

| Field | Type | Description |
|-------|------|-------------|
|-------|------|-------------|

AGS Wiki

[On this page](#)[Overview](#)

| | | |
|------------------|--|--|
| icons | string | Icon directory to append to Gtk.IconName. |
| windows | Array<Gtk.Window> | List of Windows . |
| closeWindowDelay | Record<string, number> | Delays the closing of a window, this is useful for making animations with a revealer |
| onConfigParsed | (app: App) => void | Callback to execute after all user modules were resolved. |
| onWindowToggled | (windowName: string, visible: boolean) => void | Callback to execute when a window is toggled. |

 [Edit page](#)[Previous](#) [Theming](#)[Next](#) [Type Checking](#)

On this page > Usage

Custom Service

Writing a custom Service is as simple as

- declaring a class
- defining its signals and properties
- declaring get/set for properties

This is an example Service for backlight using `brightnessctl` to set the brightness and `Utils.monitorFile` to watch for changes.

brightness.js

AGS Wiki



On this page > Usage

```
// the class itself
// an object defining the signals
// an object defining its properties
Service.register(
    this,
    {
        // 'name-of-signal': [type as a string from GObject.TYPE_<type>]
        'screen-changed': ['float'],
    },
    {
        // 'kebab-cased-name': [type as a string from GObject.TYPE_<type>]
        // 'r' means readable
        // 'w' means writable
        // guess what 'rw' means
        'screen-value': ['float', 'rw'],
    },
),
);

// this Service assumes only one device with backlight
#interface = Utils.exec("sh -c 'ls -wl /sys/class/backlight | head -1'");

// # prefix means private in JS
#screenValue = 0;
#max = Number(Utils.exec('brightnessctl max'));

// the getter has to be in snake_case
get screen_value() {
    return this.#screenValue;
}

// the setter has to be in snake_case too
set screen_value(percent) {
    if (percent < 0)
```

AGS Wiki



On this page > Usage

```
    Utils.execAsync(`brightnessctl set ${percent * 100}% -q`);
    // the file monitor will handle the rest
}

constructor() {
    super();

    // setup monitor
    const brightness = `/sys/class/backlight/${this.#interface}/brightness`;
    Utils.monitorFile(brightness, () => this.#onChange());

    // initialize
    this.#onChange();
}

#onChange() {
    this.#screenValue = Number(Utils.exec('brightnessctl get')) / this.#max;

    // signals have to be explicitly emitted
    this.emit('changed'); // emits "changed"
    this.notify('screen-value'); // emits "notify::screen-value"

    // or use Service.changed(propName: string) which does the above two
    // this.changed('screen-value');

    // emit screen-changed with the percent as a parameter
    this.emit('screen-changed', this.#screenValue);
}

// overwriting the connect method, let's you
// change the default event that widgets connect to
connect(event = 'screen-changed', callback) {
    return super.connect(event, callback);
}
```

On this page > Usage

```
const service = new BrightnessService;

// export to use in other modules
export default service;
```

Note

For bind to work, the property has to be defined in Service.register

Usage

Using it with widgets is as simple as using the builtin ones.

AGS Wiki



On this page Usage

```
        value: brightness.bind('screen-value'),
    });

const label = Label({
    label: brightness.bind('screen-value').transform(v => `#${v}`),
    setup: self => self.hook(brightness, (self, screenValue) => {
        // screenValue is the passed parameter from the 'screen-changed' signal
        self.label = screenValue ?? 0;

        // NOTE:
        // since hooks are run upon construction
        // the passed screenValue will be undefined the first time

        // all three are valid
        self.label = `${brightness.screenValue}`;
        self.label = `${brightness.screen_value}`;
        self.label = `${brightness['screen-value']}`;

    }, 'screen-changed'),
});
```

Edit page

← Previous
App

Next
Subclassing GTK →
Widgets

[On this page](#)[Overview](#)

Home Manager

Example content of a `flake.nix` file

```
flake.nix
```

AGS Wiki



On this page > Overview

```
url = "github:nix-community/home-manager";
inputs.nixpkgs.follows = "nixpkgs";
};

# add ags
ags.url = "github:Aylur/ags";
};

outputs = { home-manager, nixpkgs, ... }@inputs:
let
  system = "x86_64-linux";
in
{
  homeConfigurations."${username}" = home-manager.lib.homeManagerConfiguration
    pkgs = import nixpkgs { inherit system; };

    # pass inputs as specialArgs
    extraSpecialArgs = { inherit inputs; };

    # import your home.nix
    modules = [ ./home-manager/home.nix ];
};

};

}
```

Example content of `home.nix` file

```
home.nix
```

AGS Wiki



On this page > Overview

```
programs.agс = {  
    enable = true;  
  
    # null or path, leave as null if you don't want hm to manage the config  
    configDir = ../ags;  
  
    # additional packages to add to gjs's runtime  
    extraPackages = with pkgs; [  
        gtksourceview  
        webkitgtk  
        accountsservice  
    ];  
};  
}
```

Edit page

[On this page](#)[Modules](#)

JavaScript

If you don't know any JavaScript, this is a quick 5 minute course explaining most things you will need to understand.

Docs

If you need a more in depth explanation you can look up anything on developer.mozilla.org, everything except for the [dom](#) will apply to gjs.

Semicolons

Semicolons are optional, as there is [automatic semicolon insertions](#)

Logging, printing

```
// in gjs and other runtimes
console.log('hello')

// in gjs
print('hello')
```

Variables



On this page > Modules

```
let someBool = true

const array = ['string', 69, true]

const object = {
    key: 'value',
    'key-word': 'value',
}
```

Flow control

if else

```
let i = 0
if (i < 0) {
    print('its negative')
}
else if(i > 0) {
    print('its more than 0')
}
else {
    print('its 0')
}
```

while loop

AGS Wiki

[On this page](#)[Modules](#)

}

for loop

```
for (let i = 0; 0 < 5; i++) {  
    print(i)  
}
```

for of loop

```
const array = [0, 1, 2, 3, 4]  
for (const item of array) {  
    print(item)  
}
```

Functions

There are multiple ways to define functions, with the `function` keyword or with [fat arrow functions](#), also known as lambda functions in other languages

named function

```
function fn(param1, param2) {  
    print(param1, param2)  
}
```

nameless function assigned to a const variable

AGS Wiki



On this page > Modules

fat arrow function a.k.a lambda

```
const fn = (param1, param2) => {
    print(param1, param2)
}
```

invoke all of them like any other function

```
fn('hi', 'mom')
```

default parameter value

```
function fn(param1, param2 = 'hello') {
    print(param1, param2)
}

fn() // undefined, hello
```

Destructuring

arrays

```
const array = [1, 2, 3, 4, 5]
const [elem1, elem2, ...rest] = array
print(elem1, elem2, rest) // 1, 2, [3, 4, 5]
```

objects



On this page > Modules

useful in function definitions

```
function fn({ one, two }) {
    print(one, two)
}

fn({ one: 'hello', two: 'mom' }) // hello, mom
fn() // throws error because undefined can't be destructued
```

Modules

exporting

```
export default 'hello'
export const hi = 'mom'
export function fn(...params) {
    print(...params)
}
```

importing

```
import hi, { string, fn } from './path-to-file.js'
fn(hi, string) // hello, mom
```

String formatting

use backticks and \${}

AGS Wiki

[On this page](#)[Modules](#) [Edit page](#)[Previous](#) [Installation](#)[Next](#) [First Widgets](#)

[On this page](#) > [Overview](#)

Reactivity

We have used `poll` and `bind` so far to make widgets have content dynamically. There is also `on` and `hook` methods.

You can call these on any Widget that you have a reference on. They will return `this` reference, meaning you can chain them up in any order in any number.

```
const widget = Widget()
widget.hook()
widget.bind()
```

```
const widget = Widget()
    .hook()
    .bind()
```

```
const widget = Widget({
    setup: self => {
        self.bind()
        self.hook()
    }
})
```

[On this page](#)[Overview](#)

})

Hook

`hook` will setup a listener to a `GObject` and will handle disconnection when the widget is destroyed. It will connect to the `changed` signal by default when not specified otherwise.

```
// .hook(GObject, callback, signal?)  
const BatteryPercent = () => Label()  
    .hook(Battery, label => {  
        label.label = `${Battery.percent}%`  
        label.visible = Battery.available  
    }, 'changed')
```

Bind

`bind` can be directly translated to `hook`. It will setup a listener based on property changes

On this page > Overview

```
Battery, // GObject to listen to
'percent', // target property
p => `${p}%` // optional transform method

// translates to
label.hook(
  Battery,
  self => self['label'] = `${Battery['percent']}%`,
  'notify::percent')
```

It is also possible to call `bind` on [Services](#) and [Variables](#) that can be used inside constructors.

```
Label({
  label: Battery
    .bind('percent')
    .transform(p => `${p}%`)
})
```

```
const text = Variable('hello')

Label({
  label: text
    .bind()
    .transform(v => `transformed ${v}`)
})
```

On

`on` is the same as `connect` but instead of the signal handler id, it returns a reference to the

[On this page](#) > [Overview](#)

```
function MyButton() {
    const self = Widget.Button()

    self.connect('clicked', () => {
        print(self, 'is clicked')
    })

    return self
}
```

```
const MyButton = () => Widget.Button()
    .on('clicked', self => {
        print(self, 'is clicked')
    })
}
```

Poll

Avoid using this as much as possible, using this is considered bad practice.

These two are equivalent

AGS Wiki



On this page Overview

```
        self.label = Utils.exec('date')
    }, self)

    return self
}
```

```
const MyLabel = () => Widget.Label()
    .poll(1000, self => {
        self.label = Utils.exec('date')
    })
}
```

Edit page

Previous

First Widgets

Next

Theming

[On this page](#) > [Overview](#)

Services

A Service is an instance of a [GObject.Object](#) that emits signals when its state changes. Widgets can connect to them and execute a callback function on their signals which are usually functions that updates the widget's properties.

```
const widget = Widget.Label()
    // the signal is 'changed' if not specified
    // [Service, callback, signal = 'changed']
    .hook(someService, function(self, ...args) {
        // there can be other arguments based on signals
        self.label = 'new label'
    }, 'changed')

    // [prop, Service, targetProp, transfrom = out => out]
    .bind('label', SomeService, 'service-prop', function(serviceProp) {
        return `transformed ${serviceProp}`
    })
}
```

```
Widget.Label({
    label: someService.bind('service-prop').transfrom(serviceProp => {
        return `transformed ${serviceProp}`
    }),
})
```

Services can be also connected outside of widgets



On this page

Overview

⚠️ Caution

If you reference a widget inside `someService.connect`, make sure to handle disconnection as well if that widget is destroyed

List of builtin services

```
use Service.import
```

```
const battery = await Service.import('battery')
```

or if you prefer static import

```
import { battery } from 'resource:///com/github/Aylur/ags/service/battery.js';
```

⚠️ Caution

Import `default` and don't import the service class from the module, unless you need the type when using typescript.

```
// this is the service singleton instance
import Battery from 'resource:///com/github/Aylur/ags/service/battery.js';

// this is also the service singleton instance
import { battery } from 'resource:///com/github/Aylur/ags/service/battery.js'

Battery === battery // true
```

[On this page](#)[Overview](#)

Tip

Every service has a "changed" signal which is emitted on any kind of state change, unless stated otherwise.

- [applications](#)
- [audio](#)
- [battery](#)
- [bluetooth](#)
- [greetd](#)
- [hyprlnd](#)
- [mpрис](#)
- [network](#)
- [notifications](#)
- [power Profle](#)
- [system Tray](#)

[!\[\]\(4eb97b4954f316b04cb081f8336f0f6e_img.jpg\) Edit page](#)[!\[\]\(0dc270d8e0ea635410c967d2a04fc59b_img.jpg\) Previous](#)[**Variables**](#)[!\[\]\(99ac22152619eb2af41346ffe817d250_img.jpg\) Next](#)[**Utils**](#)



On this page

Custom Subclassing

Subclassing GTK Widgets

Using Gtk.Widgets not builtin

Use them like regular GTK widgets

```
import Gtk from 'gi://Gtk';

const calendar = new Gtk.Calendar({
    showDayNames: false,
    showHeading: true,
});
```

You can subclass Gtk.Widget not builtin to behave like AGS widgets.

```
const Calendar = Widget.subclass(Gtk.Calendar)

const myCalendar = Calendar({
    showDayNames: false,
    showHeading: true,

    // now you can set AGS props
    className: 'my-calendar',
    setup(self) {
        self.bind()
    }
});
```



Note

Open up an issue/PR if you want to see a widget to be available on `Widget` by default.

Custom Subclassing

Usually in GTK custom widgets are achieved by subclassing. The idea behind AGS is to use functions that create widgets and utilize [closures](#).

```
function CounterButton(({ color = 'aqua', ...rest })) {
    const count = Variable(0);

    const label = Widget.Label({
        label: count.bind().transform(v => `count: ${v}`),
        style: `color: ${color}`,
    });

    return Widget.Button({
        ...rest, // spread passed parameters
        child: label,
        onClicked: () => count++,
    })
}

// then simply call it
const button = CounterButton({
    color: 'blue',
    className: 'my-widget',
});
```

AGS Wiki

[On this page](#)[Custom Subclassing](#)



On this page > Custom Subclassing

```
        'count': ['int', 'rw']
    }
})

}

// the super constructor will take care of setting the count prop
// so you don't have to explicitly set count in the constructor
constructor(props) {
    super(props);

    const label = new Gtk.Label({
        label: `${this.count}`,
    });

    this.add(label);

    this.connect('clicked', () => {
        this.count++;
        label.label = `${this.count}`;
    });
}

get count() {
    return this._count || 0;
}

set count(num) {
    this._count = num;
    this.notify('count');
}
}
```



On this page Custom Subclassing

```
counter.○,  
  
    // you can set AGS widget props on it  
    className: '',  
)  
  
counterButton.connect('notify::count', ({ count }) => {  
    print(count);  
})
```

Edit page

Previous

← Custom Service

Next

→ Examples



On this page Autoreload Css

Theming

Since the widget toolkit is **GTK3** theming is done with **CSS**.

- [CSS tutorial](#)
- [GTK CSS Overview wiki](#)
- [GTK CSS Properties Overview wiki](#)



GTK is not the web

While most features are implemented in GTK, you can't assume anything that works on the web will work with GTK. Refer to the [GTK docs](#) to see what is available.

So far every widget you made used your default GTK3 theme. To make them more custom, you can apply stylesheets to them, which are either imported `css` files or `inline css` applied with the `css` property.

From file at startup

config.js



On this page > Autoreload Css

```
// or relative to config.js
style: './style.css',
}
```

Css Property on Widgets

```
Widget.Label({
    css: 'color: blue; padding: 1em;',
    label: 'hello'
})
```

Apply Stylesheets at Runtime



Caution

`App.applyCss` will apply over other stylesheets applied before. You can reset stylesheets with `App.resetCss`

```
App.resetCss() // reset if need
App.applyCss('/full/path/to/file.css')
```

Inspector

If you are not sure about the widget hierarchy or any CSS selector, you can use the [GTK inspector](#)

On this page > Autoreload Css

Using pre-processors like SCSS

```
config.js

// main scss file
const scss = `${App.configDir}/style.scss`

// target css file
const css = `${App.configDir}/style.css`

// make sure sassc is installed on your system
Utils.exec(`sassc ${scss} ${css}`)

export default {
    style: css,
    windows: [ /* windows */ ],
}
```

Autoreload Css

[On this page](#) > Autoreload Css

```
// reload function
function() {
    // main scss file
    const scss = `${App.configDir}/style.scss`

    // target css file
    const css = `${App.configDir}/style.css`

    // compile, reset, apply
    Utils.exec(`sassc ${scss} ${css}`)
    App.resetCss()
    App.applyCss(css)
},
// specify that its a directory, to make the monitor recursive
'directory',
)
```

 [Edit page](#)[Previous](#) [Reactivity](#)[Next](#) [Config Object](#)



On this page

Overview

Type Checking

To have auto suggestions and type checking while working on the configuration, you will need to setup a TypeScript LSP in your IDE.



Caution

Bluetooth doesn't have type definitions yet.

Use the `--init` cli flag that will setup a `tsconfig.ts` and symlink the installed type definitions

```
ags --init  
ags --init --config /path/to/config.js
```

If you don't want typechecking only suggestions in js files unset it in `tsconfig.json`

```
"checkJs": false
```

Using TypeScript

If you want to use TypeScript, you will need to handle the build step yourself.

Here is an example using `bun build`

```
config.js
```

AGS Wiki



On this page › Overview

```
await Utils.execAsync([
  'bun', 'build', entry,
  '--outdir', outdir,
  '--external', 'resource:///*',
  '--external', 'gi:///*',
])
} catch (error) {
  console.error(error)
}

const main = await import(`file://${outdir}/main.js`)

export default main.default
```

ts/main.ts

```
const Bar = (monitor: number) => Widget.Window({
  name: `bar-${monitor}`,
  child: Widget.Label('hello'),
})

export default {
  windows: [Bar(0)]
}
```

Edit page

Previous

← Config Object

Next

→ CLI

AGS Wiki

[On this page](#)[Overview](#)

Utils

Running external commands

Synchronously

```
function exec(cmd: string): string
```

This is synchronous, meaning it will **block the eventloop**

```
const echo = Utils.exec('echo "Hi Mom"') // returns string
console.log(echo) // logs "Hi Mom"
```

```
const uptime = Utils.exec(`bash -c "uptime | awk '{print $3}' | tr ',' ' ' `)
console.log(uptime)
```

Asynchronously

```
function execAsync(cmd: string | string[]): Promise<string>;
```

This won't block,

On this page

> Authentication

Note

Both `exec` and `execAsync` launches the given binary on its own, meaning if you want to use `|` pipes or any other shell features then you have to run it with bash.

```
Utils.execAsync(['bash', '-c', 'cmd | cmd && cmd']);
Utils.exec('bash -c "cmd | cmd && cmd"');
```

Running external scripts

```
function subprocess(
    cmd: string | string[],
    callback: (out: string) => void,
    onError = logError,
    bind?: Gtk.Widget,
): Gio.Subprocess
```

Takes two to four arguments, returns [Gio.Subprocess](#)

On this page

> Authentication

```
// callback when the program outputs something to stdout
(output) => print(output),

// callback on error
(err) => logError(err),

// optional widget parameter
// if the widget is destroyed the subprocess is forced to quit
widget,
)
```

Killing the process

```
proc.force_exit()
```

Writing and reading files

```
function readFile(file: string | Gio.File): string
function readFileSync(file: string | Gio.File): Promise<string>
```

Synchronously reading, returns a string

```
const contents = Utils.readFile('path-to-file')
```

Asynchronously reading, returns a Promise

On this page > Authentication

Asynchronously writing, returns a Promise

```
Utils.writeFile('Contents: Hi Mom', 'path-to-file')
  .then(file => print('file is the Gio.File'))
  .catch(err => print(err))
```

Monitoring files and directories

```
function monitorFile(
  path: string,
  callback?: (file: Gio.File, event: Gio.FileMonitorEvent) => void,
  type: 'file' | 'directory' = 'file',
  flags = Gio.FileMonitorFlags.NONE,
): Gio.FileMonitor | null
```

```
const monitor = Utils.monitorFile('/path/to/file', (file, event) => {
  print(Utils.readFile(file), event)
})
```

Cancelling the monitor

```
monitor.cancel()
```

Timeout and Interval

AGS Wiki



On this page > Authentication

```
const source = setTimeout(() => { /* callback */ }, 1000)
```

Interval

```
const source = setInterval(() => { /* callback */ }, 1000)
```

To cancel them use `GLib.Source.destroy`

```
source.destroy()
```

You can use the ones from `Utils`

```
function interval(
  interval: number,
  callback: () => void,
  bind?: Gtk.Widget,
): number
```

```
function timeout(
  ms: number,
  callback: () => void,
): number
```

```
const id = Utils.timeout(1000, () => {
  // runs with a second delay
})
```

The widget parameter is optional



On this page > Authentication

If you pass a widget to `Utils.interval`, it will automatically be canceled, when the widget is destroyed

```
const widget = Widget.Label()
const id = Utils.interval(1000, () => {}, label)
widget.destroy()
```

To cancel them use `GLib.source_remove`

```
import GLib from 'gi://GLib'
GLib.source_remove(id)
```

Lookup an Icon name

```
const icon = Utils.lookUpIcon('dialog-information-symbolic')

if (icon) {
    // icon is the corresponding Gtk.IconInfo
}
else {
    // null if it wasn't found in the current Icon Theme
}
```

Fetch

should be pretty close to the web api

On this page > Authentication

Authentication

authenticate a user using pam

Note

on NixOS make sure you have `security.pam.services.agss = {}` in `configuration.nix`

```
Utils.authenticate('password')
  .then(() => print('authentication sucessful'))
  .catch(err => logError(err, 'unsucessful'))

Utils.authenticateUser("username", "password")
  .then(() => print("authentication sucessful"))
  .catch(err => logError(err, 'unsucessful'))
```

Send Notifications

```
Utils.notify('summary', 'body', 'icon-name')
```

AGS Wiki



On this page Authentication

```
actions: {  
    'Click Me': () => print('clicked')  
}  
})  
  
const notifications = await Service.import('notifications')  
const n = notifications.getNotification(id)
```

Edit page

Previous

Services

Next

App



On this page > Example RAM and CPU usage

Variables

Variable is just a simple `GObject` that holds a value.



Tip

The global `Variable` is a function that returns an instance If you want to subclass this instead of `Service` import the class

```
import { Variable as Var } from 'resource:///com/github/Aylur/ags/variable.js'
class MyVar extends Var {
    static { Service.register(this) }
}
```

Polling and Listening to executables

AGS Wiki



On this page > Example RAM and CPU usage

```
listen: ['bash', '-c', 'some-command'],  
  
// can also take a transform function  
listen: [App.configDir + '/script.sh', out => JSON.parse(out)],  
listen: [['bash', '-c', 'some-command'], out => JSON.parse(out)],  
  
// poll is a [interval: number, cmd: string[] | string, transform: (string)  
// cmd is what gets passed to Utils.execAsync  
poll: [1000, 'some-command'],  
poll: [1000, 'some-command', out => 'transformed output: ' + out],  
poll: [1000, ['bash', '-c', 'some-command'], out => 'transformed output: ' +  
  
// or [number, function]  
poll: [1000, () => { return new Date(); }],  
poll: [1000, Math.random],  
});
```

Updating its value

```
myVar.value = 'new-value'  
myVar.setValue('new-value')
```

Getting its value

```
print(myVar.value)  
print(myVar.getValue())
```

Temporarily stopping it



On this page > Example RAM and CPU usage

Starting it

It will start on construction, no need to explicitly call this.

```
variable.startListen() // launches the subprocess again  
variable.startPoll()
```

Getting if its active

```
print(variable.isListening)  
print(variable.isPolling)
```

Usage with widgets

```
const label = Widget.Label({  
    label: myVar.bind(),  
  
    // optional transform method  
    label: myVar.bind().transform(value => value.toString()),  
  
    // hook to do more than an assignment on changed  
    setup: self => self.hook(myVar, () => {  
        self.label = myVar.value.toString();  
    })  
})
```

Connecting to it directly

[On this page](#)[Example RAM and CPU usage](#)

Dispose if no longer needed

This will stop the interval and force exit the subprocess

```
myVar.dispose();
```

Example RAM and CPU usage

AGS Wiki



On this page Example RAM and CPU usage

```
.find(line => line.includes('Cpu(s)'))  
.split(/\s+/)[1]  
.replace(',', ', .')]],  
)  
  
const ram = Variable(0, {  
  poll: [2000, 'free', out => divide(out.split('\n')  
.find(line => line.includes('Mem:'))  

```

Edit page

← Previous
Widgets

Next →
Services



On this page > CircularProgress

Widgets

Widget functions return an instance of [Gtk.Widget](#). Most common widgets are subclassed and have a few additional properties.

These widgets have some additional properties on top of the base Gtk.Widget ones:

| Property | Type | Description |
|-------------|-----------------|--|
| class-name | string | List of class CSS selectors separated by white space. |
| class-names | Array< string > | List of class CSS selectors. |
| css | string | Inline CSS. e.g <code>label { color: white; }</code> . If no selector is specified <code>*</code> will be assumed. e.g <code>color: white;</code> will be inferred as <code>*{ color: white; }</code> . |
| hpack | string | Horizontal alignment, behaves like <code>halign</code> . <code>halign</code> takes an enum, but for convenience <code>hpack</code> can be given with a string, so one of <code>"start"</code> , <code>"center"</code> , <code>"end"</code> , <code>"fill"</code> . |
| vpack | string | Vertical alignment. |
| cursor | string | Cursor style when hovering over widgets that have hover states, e.g it won't work on labels. list of valid values . |
| attribute | any | Any additional attributes on <code>self</code> |
| setup | (self) | A callback function to execute on <code>self</code> |

AGS Wiki



On this page > CircularProgress

Some common Gtk.Widget properties you might want for example

| Property | Type | Description |
|--------------|---------|---|
| hexpand | boolean | Expand horizontally. |
| vexpand | boolean | Expand vertically. |
| sensitive | boolean | Makes the widget interactable. |
| tooltip-text | string | Tooltip popup when the widget is hovered over. |
| visible | boolean | Visibility of the widget. Setting this to <code>false</code> doesn't have any effect if the parent container calls <code>show_all()</code> , for example when you set a Box's children dynamically. |

If you don't want to mutate the `classNames` array, there is `toggleClassName : (name: string, enable: boolean) => void`

```
const label = Widget.Label('example-label')

// add class name
label.toggleClassName('my-awesome-label', true)

// remove class name
label.toggleClassName('my-awesome-label', false)
```

The properties listed here are just the additional properties on top of their base Gtk.Widget classes. [Refer to the Gtk3 docs](#) for the rest of them.

Window

AGS Wiki



On this page

CircularProgress

| child | Widget | |
|-------------|-------------|---|
| name | string | Name of the window. This has to be unique, if you pass it in config. This will also be the name of the layer. |
| anchor | string[] | Valid values are "top" , "bottom" , "left" , "right" . Anchor points of the window. Leave it empty to make it centered. |
| exclusivity | string | Specify if the compositor should reserve space for the window automatically or how the window should interact with windows that do. Possible values: exclusive (space should be reserved), normal (the window should move if occluding another), ignore (the window should not be moved). Default: normal . |
| layer | string | Valid values are "overlay" , "top" , "bottom" , "background" . It is "top" by default. |
| margins | number[] | Corresponds to CSS notation, e.g [0, 6] would have 0 margin on the top and bottom and would have 6 on the right and left. |
| monitor | number | Which monitor to show the window on. If it is left undefined the window will show on the currently focused monitor. |
| popup | boolean | Pressing <code>ESC</code> while the window has focus will close it. |
| keymode | string | Valid values are "none" , "on-demand" : can receive keyboard input if focused, "exclusive" : steal keyboard input on top and overlay layers |
| gdkmonitor | Gdk.Monitor | alternative to <code>monitor</code> |



On this page > CircularProgress

```
keymode: 'on-demand',
layer: 'top',
margin: [0, 6],
monitor: 0,
child: Widget.Label('hello'),
})
```

Box

subclass of [Gtk.Box](#)

| Property | Type | Description |
|----------|----------|--|
| vertical | bool | setting vertical: true is the same as orientation: 1 |
| children | Widget[] | List of child widgets. |

On this page > CircularProgress

```
// set children dynamically
setup: self => self.hook(SomeService, () => {
    self.children = [
        /* widgets */
    ]

    // or if you want to add one child
    self.add(/* widget */)
    self.show_all() // widgets are invisible by default
}),
})
```

Button

subclass of [Gtk.Button](#)

| Property | Type |
|----------------------------|-------------------------------|
| child | Widget |
| on-clicked | () => void |
| on-primary-click | (event: Gdk.Event) => boolean |
| on-secondary-click | (event: Gdk.Event) => boolean |
| on-middle-click | (event: Gdk.Event) => boolean |
| on-primary-click-release | (event: Gdk.Event) => boolean |
| on-secondary-click-release | (event: Gdk.Event) => boolean |
| on-middle-click-release | (event: Gdk.Event) => boolean |

AGS Wiki

[On this page](#)[CircularProgress](#)

| | |
|----------------|-------------------------------|
| on-hover-lost | (event: Gdk.Event) -> boolean |
| on-scroll-up | (event: Gdk.Event) => boolean |
| on-scroll-down | (event: Gdk.Event) => boolean |

```
const button = Widget.Button({
    child: Widget.Label('click me!'),
    onPrimaryClick: () => print('echo hello'),
})
```

CenterBox

subclass of Box

| Property | Type |
|---------------|------------|
| start-widget | Gtk.Widget |
| center-widget | Gtk.Widget |
| end-widget | Gtk.Widget |

```
const centerBox = Widget.CenterBox({
    spacing: 8,
    vertical: false,
    startWidget: Widget.Label('left widget'),
    centerWidget: Widget.Label('center widget'),
    endWidget: Widget.Label('right widget'),
})
```

AGS Wiki



On this page

CircularProgress

| Property | Type | Description |
|----------|---------|---|
| start-at | number | Number between 0 and 1, e.g 0.75 is the top |
| end-at | number | Number between 0 and 1 |
| inverted | boolean | |
| rounded | boolean | Whether the progress bar should have rounded ends |
| value | number | Number between 0 and 1 |

```
const progress = Widget.CircularProgress({
    style:
        'min-width: 50px;' + // its size is min(min-height, min-width)
        'min-height: 50px;' +
        'font-size: 6px;' + // to set its thickness set font-size on it
        'margin: 4px;' + // you can set margin on it
        'background-color: #131313;' + // set its bg color
        'color: aqua;', // set its fg color

    value: battery.bind('percent').transform(p => p / 100),
    child: Widget.Icon({
        icon: battery.bind('icon-name'),
    }),
    rounded: false,
    inverted: false,
    startAt: 0.75,
})
```

Entry

subclass of [Gtk.Entry](#)

AGS Wiki



On this page > CircularProgress

on-accept () => void

```
const entry = Widget.Entry({  
    placeholder_text: 'type here',  
    text: 'initial text',  
    visibility: true, // set to false for passwords  
    onAccept: ({ text }) => print(text),  
})
```

EventBox

subclass of [Gtk.EventBox](#)

| Property | Type |
|----------------------------|-------------------------------|
| child | Widget |
| on-primary-click | (event: Gdk.Event) => boolean |
| on-secondary-click | (event: Gdk.Event) => boolean |
| on-middle-click | (event: Gdk.Event) => boolean |
| on-primary-click-release | (event: Gdk.Event) => boolean |
| on-secondary-click-release | (event: Gdk.Event) => boolean |
| on-middle-click-release | (event: Gdk.Event) => boolean |
| on-hover | (event: Gdk.Event) => boolean |
| on-hover-lost | (event: Gdk.Event) => boolean |
| on-scroll-up | (event: Gdk.Event) => boolean |

On this page > CircularProgress

Icon

subclass of [Gtk.Image](#)

| Property | Type | Description |
|----------|--------|-----------------------------------|
| icon | string | Name of an icon or path to a file |
| size | number | Forced size |

```
Widget.Icon({ icon: 'dialog-information-symbolic' })

// if you only want an icon, it can be shortened like this
Widget.Icon('dialog-information-symbolic')

// if you don't set a size, it will be computed from css font-size
Widget.Icon({
    icon: 'dialog-information-symbolic',
    style: 'font-size: 30px',
})

// NOTE:
// setting the icon dynamically has a flicker currently
// to fix it, use a forced size
Widget.Icon({
    icon: 'dialog-information-symbolic',
    size: 30,
})
```

Label

AGS Wiki



On this page

> CircularProgress

| | | |
|---------------|--------|---|
| justification | string | Valid values are "left" , "center" , "right" , "fill" . Same as justify but represented as a string instead of enum. |
| truncate | string | Valid values are "none" , "start" , "middle" , "end" . Same as ellipsize but represented as a string instead of enum. |

```
const label = Widget.Label({  
    label: 'some text to show',  
    justification: 'left',  
    truncate: 'end',  
    xalign: 0,  
    maxWidthChars: 24,  
    wrap: true,  
    useMarkup: true,  
})
```

Overlay

subclass of [Gtk.Overlay](#) Takes the size of its first child, then places subsequent children on top of each other and won't render them outside the container.

| Property | Type | Description |
|--------------|----------|--|
| child | Widget | Child which will determine the size of the overlay |
| overlays | Widget[] | Overlaid children |
| pass-through | boolean | Whether the overlay childs should pass the input through |

ProgressBar



On this page

CircularProgress

| | | |
|----------|--------|--|
| vertical | bool | Setting <code>vertical: true</code> is the same as <code>orientation: 1</code> |
| value | number | Same as <code>ProgressBar.fraction</code> |

Revealer

subclass of [Gtk.Revealer](#)

| Property | Type | Description |
|------------|--------|---|
| child | Widget | |
| transition | string | Valid values are <code>"none"</code> , <code>"crossfade"</code> , <code>"slide_left"</code> , <code>"slide_right"</code> , <code>"slide_down"</code> , <code>"slide_up"</code> . This is <code>transitionType</code> represented as a string instead of enum. |

```
const revealer = Widget.Revealer({
    revealChild: false,
    transitionDuration: 1000,
    transition: 'slide_right',
    child: Widget.Label('hello!'),
    setup: self => self.poll(2000, () => {
        self.reveal_child = !self.reveal_child;
    }),
})
```

ScrolledWindow

subclass of [Gtk.ScrolledWindow](#)



On this page CircularProgress

| | | |
|---------|--------|--|
| hscroll | string | Valid values are "always", "automatic", "external", "never". |
| vscroll | string | Valid values are "always", "automatic", "external", "never". |

```
const scrollable = Widget.Scrollable({
    hscroll: 'always',
    vscroll: 'never',
    style: 'min-width: 20px;',
    child: Widget.Label('Lorem ipsum dolor sit amet, ' +
        'officia excepteur ex fugiat reprehenderit enim ' +
        'labore culpa sint ad nisi Lorem pariatur mollit'),
})
```

Slider

subclass of [Gtk.Scale](#)

| Property | Type | Description |
|-----------|-----------------------|---|
| vertical | bool | Setting <code>vertical: true</code> is the same as <code>orientation: 1</code> |
| value | number | |
| min | number | |
| max | number | |
| marks | tuple or number | where tuple is [number, string?, Position?], Position is "top", "left", "right", "bottom" |
| on-change | (event: Gdk.Event) => | |

AGS Wiki



On this page > CircularProgress

```
Widget.Slider({
    onChange: ({ value }) => print(value),
    vertical: true,
    value: 0,
    min: 0,
    max: 1,
    marks: [
        1,
        2,
        [3, 'label'],
        [4, 'label', 'bottom'],
    ],
})
```

Stack

subclass of [Gtk.Stack](#)

| Property | Type | Description |
|----------|------------------|---|
| items | [string, Widget] | name - Widget pairs |
| shown | string | Name of the widget to show. This can't be set on construction, instead the first give widget will be shown. |

AGS Wiki

[On this page](#)[CircularProgress](#)

transition

string

slide_left_right , slide_up_down , over_up , over_down ,
over_left , over_right , under_up , under_down , under_left ,
under_right , over_up_down , over_down_up , over_left_right ,
over_right_left

```
const stack = Widget.Stack({  
    items: [  
        ['child1', Widget.Label('first child')],  
        ['child2', Widget.Label('second child')],  
    ],  
    setup: self => self.hook(gobject, () => {  
        self.shown = 'child2';  
    })  
})
```

Menu

subclass of [Gtk.Menu](#)

| Property | Type |
|----------------|--|
| children | MenuItem[] |
| on-popup | (flipped_rect: void, final_rect: void, flipped_x: boolean, flipped_y: boolean) => void |
| on-move-scroll | (scroll_type: Gtk.ScrollType) => void |



On this page CircularProgress

```
children: [
    Widget.MenuItem({
        child: Widget.Label('hello'),
    }),
],
}).popup_at_pointer(event),
})
```

MenuItem

subclass of [Gtk.MenuItem](#)

| Property | Type | Description |
|-------------|---------------|-------------|
| child | Widget | |
| on-activate | () => boolean | |
| on-select | () => boolean | |
| on-deselect | () => boolean | |

Edit page

← Previous

CLI

Next →

Variables



On this page

> Signals

Your First Widget

Start by creating `~/.config/ags/config.js` with the following contents:

```
~/.config/ags/config.js

export default {
  windows: [
    // this is where window definitions will go
  ]
}
```

then run `ags` in the terminal

```
ags
```

You will see nothing happen, just AGS hanging, this is because you have an empty config. Running `ags` will execute the config like a regular script, it is just a library over GTK, and its on **you** to program your windows and widgets.

Tip

run `ags --init` to generate a `tsconfig.json` and **symlink** the generated types, that will provide typesafety and autosuggestions



On this page > Signals

```
const myLabel = Widget.Label({  
    label: 'some example content',  
})  
  
const myBar = Widget.Window({  
    name: 'bar',  
    anchor: ['top', 'left', 'right'],  
    child: myLabel,  
})  
  
export default { windows: [myBar] }
```

Tip

GObject properties can be accessed or set in multiple ways: with `camelCase`, `snake_case`, and `kebab-case`

```
const w = Widget({  
    className: 'my-label',  
    class_name: 'my-label',  
    'class-name': 'my-label',  
})  
  
w.className = ''  
w.class_name = ''  
w['class-name'] = ''  
  
w['className'] = ''  
w['class_name'] = ''
```



On this page > Signals

~~You have two monitors and want to have a bar for each. Make a function that returns a~~

Gtk.Widget instance.

```
function Bar(monitor = 0) {
    const myLabel = Widget.Label({
        label: 'some example content',
    })

    return Widget.Window({
        monitor,
        name: `bar${monitor}`, // this name has to be unique
        anchor: ['top', 'left', 'right'],
        child: myLabel,
    })
}

export default {
    windows: [
        Bar(0), // can be instantiated for each monitor
        Bar(1),
    ],
}
```

Note

The `name` attribute only has to be unique, if it is passed to `windows` in the exported object.

Calling `Widget.Window` will create and show the window by default. It is not necessary to pass a reference to `windows` in the exported object, but if it is not, it can't be toggled with `ags --toggle-window` or through `App.toggleWindow`



On this page > Signals

second window date .

```
function Bar(monitor = 0) {
    const myLabel = Widget.Label({
        label: 'some example content',
    })

    Utils.interval(1000, () => {
        myLabel.label = Utils.exec('date')
    })

    return Widget.Window({
        monitor,
        name: `bar${monitor}`,
        anchor: ['top', 'left', 'right'],
        child: myLabel,
    })
}
```

Note

JavaScript is **single threaded** and `exec` is a **blocking operation**, for a `date` call it's fine, but usually you want to use its **async** version: `execAsync`.

Looking great, but that code has too much boilerplate. Let's use a fat arrow instead of the `function` keyword, and instead of calling `interval` let's use the `poll` method.



On this page > Signals

```
    child: Widget.Label()
        .poll(1000, label => label.label = exec('date')),
    })
```

Tip

That is still not the best solution, because when you instantiate multiple instances of `Bar` each will call `exec` separately. What you want to do is move the date into a `Variable` and bind it.

```
const date = Variable('', {
    poll: [1000, 'date'],
})

const Bar = () => Widget.Window({
    child: Widget.Label({ label: date.bind() })
})
```

Signals

Usually it is best to avoid polling. Rule of thumb: the less intervals the better. This is where `GObject` shines. We can use **signals** for pretty much everything.

anytime `myVariable.value` changes it will send a signal and things can react to it

```
const myVariable = Variable(0)
```

for example execute a callback



On this page > Signals

bind its value to a widget's property

```
const bar = Widget.Window({  
    name: 'bar',  
    child: Widget.Label({  
        label: myVariable.bind().transform(v => `value: ${v}`)  
    }),  
})
```

incrementing the `value` causes the label to update and the callback to execute

```
myVariable.value++
```

For example with `pactl` it is possible to query information about the volume level, but we don't want to have an interval that checks it periodically. We want a **signal** that signals every time its **changed**, so that we only do operations when its needed, and therefore we don't waste resources. `pactl subscribe` writes to stdout everytime there is a **change**.

On this page > Signals

```
    }],
})

pactl.connect('changed', ({ value }) => {
    print(value.msg, value.count)
})

const label = Widget.Label({
    label: pactl.bind().transform(({ count, msg }) => {
        return `${msg} ${count}`
    )),
})

// widgets are GObjects too
label.connect('notify::label', ({ label }) => {
    print('label changed to ', label)
})
```

Avoiding external scripts

For *most* of your system, you don't have to use external scripts and binaries to query information. AGS has builtin [Services](#). They are just like [Variables](#) but instead of a single `value` they have more attributes and methods on them.

AGS Wiki

[On this page](#)[Signals](#)

```
child: Widget.Icon({  
    icon: battery.bind('icon_name'),  
}),  
})
```

 [Edit page](#)[Previous](#)[← JavaScript](#)[Next](#)[Reactivity →](#)

API Guide

lib

This section covers important functions available in Flakelight's lib attribute.

mkFlake

The outputs of a flake using Flakelight are created using the `mkFlake` function. When called directly, Flakelight invokes `mkFlake`, as follows:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      # Your flake configuration here
    };
}
```

To call `mkFlake` explicitly, you can do:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight.lib.mkFlake ./ {
      # Your flake configuration here
    };
}
```

`mkFlake` takes two parameters: the path to the flake's source and a Flakelight module.

If you need access to module args, you can write it as bellow:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ ({ lib, config, ... }: {
      # Your flake configuration here
    });
}
```

Module arguments

The following module arguments are available:

- `src`: The flake's source directory
- `lib`: nixpkgs lib attribute
- `config`: configured option values
- `options`: available options

- `flakelight`: flakelight lib attribute
- `inputs`: value of inputs option
- `outputs`: resulting output (i.e. final flake attributes)
- `pkgsFor`: attrset mapping systems to the pkgs set for that system
- `moduleArgs`: All of the above arguments (passed to auto-loaded files)

Additional pkgs values

Functions that take the package set as an argument, such as package definitions or `perSystem` values, have several additional values available in the package set.

The `src`, `flakelight`, `inputs`, `outputs`, and `moduleArgs` attributes are the same as the above module arguments.

`inputs'` and `outputs'` are transformed versions of `inputs` and `outputs` with system preselected. I.e., `inputs.emacs-overlay.packages.x86_64-linux.default` can be accessed as `inputs'.emacs-overlay.packages.default`.

`defaultMeta` is a derivation meta attribute set generated from options. Modules setting `packages.default` should use this to allow meta attributes to be configured.

Module options

This section covers the options available to modules.

inputs

Type: `AttrsOf FlakeInput`

The `inputs` option allows setting the flake inputs used by modules. To set the nixpkgs used for building outputs, you can pass your flake inputs in as follows:

```
{
  inputs = {
    nixpkgs.url = "nixpkgs/nixpkgs-unstable";
    flakelight.url = "github:nix-community/flakelight";
  };
  outputs = { flakelight, ... }@inputs:
    flakelight ./ {
      inherit inputs;
    };
}
```

Or to just pass just the nixpkgs input:

```
{
  inputs = {
    nixpkgs.url = "nixpkgs/nixpkgs-unstable";
    flakelight.url = "github:nix-community/flakelight";
```

```

};

outputs = { flakelight, nixpkgs, ... }:
  flakelight ./ {
    inputs.nixpkgs = nixpkgs;
  };
}

```

systems

Type: [SystemStr]

The `systems` option sets which systems per-system outputs should be created for.

If not set, the default is `x86_64-linux` and `aarch64-linux`.

To also support `i686-linux` and `armv7l-linux`, you would configure `systems` as follows:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      systems = [ "x86_64-linux" "aarch64-linux" "i686-linux" "armv7l-linux" ];
    };
}
```

To support all systems supported by flakes, set `systems` as follows:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ ({ lib, ... }): {
      systems = lib.systems.flakeExposed;
    });
}
```

To support all Linux systems supported by flakes, set `systems` as follows:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ ({ lib, ... }): {
      systems = lib.intersectLists
        lib.systems.doubles.linux
        lib.systems.flakeExposed;
    });
}
```

nixDir

Type: Path

The `nixDir` option is `./nix` by default and sets which directory to use to automatically load nix files to configure flake options from.

For a given option, the following is checked in order:

- If `${nixDir}/option.nix` exists, it is imported as the value
- Else if `${nixDir}/option` is a directory with a `default.nix`, it is imported
- Else if `${nixDir}/option` is a directory, it results in an attrset with an attr for each importable item in the directory for which the values are the corresponding items imported. An importable item is a file ending with `.nix` or a directory containing a `default.nix`.

To enable using a directory for an attrset that includes a `default` attribute, attr names can be escaped with an underscore. For example, `${nixDir}/nix/packages/_default.nix` will be loaded as `packages.default`.

Aliases for options can be set with the `nixDirAliases.nixosConfigurations` option. For example, by default `nixDirAliases.nixosConfigurations = ["nixos"];` is set which means “nixos” can be used instead of “nixosConfigurations” for loading the files as described above.

All options except for `nixDir` and `_module` can be configured this way.

outputs

Type: AttrSet | (ModuleArgs -> AttrSet)

The `outputs` option allows you to directly configure flake outputs. This should be used for porting or for configuring output attrs not otherwise supported by Flakelight.

The option value may be an attrset or a function that takes `moduleArgs` and returns an attrset.

To add a `example.test` output to your flake you could do the following:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      outputs = {
        example.test = "hello";
      };
    };
}
```

With the above, `nix eval .#example.test` will output “hello”.

This can be used to configure any output, for example directly setting an overlay (though this can be configured with the `overlays` option):

```
{  
    inputs.flakelight.url = "github:nix-community/flakelight";  
    outputs = { flakelight, ... }:  
        flakelight ./ {  
            outputs.overlays.clang = final: prev: { stdenv = final clangStdenv; };  
        };  
}
```

perSystem

Type: `Pkgs -> AttrSet`

The `perSystem` option allows you to directly configure per-system flake outputs, and gives you access to packages. This should be used for porting or for configuring output attrs not otherwise supported by Flakelight.

To add `example.${system}.test` outputs to your flake, you could do the following:

```
{  
    inputs.flakelight.url = "github:nix-community/flakelight";  
    outputs = { flakelight, ... }:  
        flakelight ./ {  
            perSystem = pkgs: {  
                example.test = pkgs.writeShellScript "test" "echo hello";  
            };  
        };  
}
```

The above, with default systems, will generate `example.x86_64-linux.test` and `example.aarch64-linux.test` attributes.

nixpkgs.config

Type: `AttrSet`

This allows you to pass configuration options to the Nixpkgs instance used for building packages and calling `perSystem`.

For example, to allow building broken or unsupported packages, you can set the option as follows:

```
{  
    inputs.flakelight.url = "github:nix-community/flakelight";  
    outputs = { flakelight, ... }:  
        flakelight ./ {  
            nixpkgs.config = { allowBroken = true; allowUnsupportedSystem = true; };  
        };  
}
```

```
    };
```

withOverlays

Type: [Overlay] | Overlay

This allows you to apply overlays to the Nixpkgs instance used for building packages and calling perSystem.

It can be set to either a list of overlays or a single overlay.

For example, to apply the Emacs overlay and change the Zig version, you can set the option as follows:

```
{
  inputs = {
    flakelight.url = "github:nix-community/flakelight";
    emacs-overlay.url = "github:nix-community/emacs-overlay";
  };
  outputs = { flakelight, emacs-overlay, ... }:
    flakelight ./ {
      withOverlays = [
        emacs-overlay.overlays.default
        (final: prev: { zig = final.zig_0_9; })
      ];
    };
}
```

You can use the values from the overlays with other options:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      withOverlays = final: prev: { testValue = "hi"; };

      package = { writeShellScript, testValue }:
        writeShellScript "test" "echo ${testValue}";
    };
}
```

packages

Types:

```
  package: PackageDef
  packages: (AttrsOf PackageDef) | (ModuleArgs -> (AttrsOf PackageDef))
  pname: Str
```

The `package` and `packages` options allow you to add packages. These are exported in the `packages.${system}` outputs, are included in `overlays.default`, and have build checks in `checks.${system}`.

`package` can be set to a package definition, and will set `packages.default`.

`packages` can be set to attrs of package definitions.

By default, the `packages.default` package's name (its attribute name in the package set and overlay) is automatically determined from the derivation's `pname`. In order to use a different attribute name from the package `pname`, to set it in cases where it cannot be automatically determined, or to speed up uncached evaluation, the flakelight `pname` option can be set.

To set the default package, you can set the options as follows:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      package = { stdenv }:
        stdenv.mkDerivation {
          pname = "pkg1";
          version = "0.0.1";
          src = ./;
          installPhase = "make DESTDIR=$out install";
        };
    };
}
```

The above will export `packages.${system}.default` attributes, add `pkg1` to `overlays.default`, and export `checks.${system}.packages-default`.

You can also instead just directly set `packages.default`.

To set multiple packages, you can set the options as follows:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      packages = {
        default = { stdenv }:
          stdenv.mkDerivation {
            name = "pkg1";
            src = ./;
            installPhase = "make DESTDIR=$out install";
          };
        pkg2 = { stdenv, pkg1, pkg3 }:
          stdenv.mkDerivation {
```

```

        name = "hello-world";
        src = ./pkg2;
        nativeBuildInputs = [ pkg1 pkg3 ];
        installPhase = "make DESTDIR=$out install";
    };
    pkg3 = { stdenv }:
        stdenv.mkDerivation {
            name = "hello-world";
            src = ./pkg3;
            installPhase = "make DESTDIR=$out install";
        };
    };
}

```

The above will export `packages.${system}.default`, `packages.${system}.pkg2`, `packages.${system}.pkg3` attributes, add `pkg1`, `pkg2`, and `pkg3` to `overlays.default`, and export corresponding build checks.

To use the first example, but manually specify the package name:

```
{
    inputs.flakelight.url = "github:nix-community/flakelight";
    outputs = { flakelight, ... }:
        flakelight ./ {
            pname = "pkgs-attribute-name";
            package = { stdenv }:
                stdenv.mkDerivation {
                    pname = "package-name";
                    version = "0.0.1";
                    src = ./;
                    installPhase = "make DESTDIR=$out install";
                };
        };
}
```

devShell

Type:

```

devShell: Cfg | (Pkgs -> Cfg) | PackageDef
Cfg.packages: [Derivation] | (Pkgs -> [Derivation])
Cfg.inputsFrom: [Derivation] | (Pkgs -> [Derivation])
Cfg.shellHook: Str | (Pkgs -> Str)
Cfg.env: (AttrsOf Str) | (Pkgs -> (AttrsOf Str))
Cfg.stdenv: Stdenv | (Pkgs -> Stdenv)

```

The `devshell` options allow you to configure `devShells.${system}.default`. It is split up into options in order to enable multiple modules to contribute to its

configuration.

`devShell` can alternatively be set to a package definition, which is then used as the default shell, overriding other options.

`devShell` can also be set to a function that takes the package set and returns an attrSet of the devShell configuration options.

The options available are as follows:

`devShell.packages` is a list of packages to add to the shell. It can optionally be a function taking the package set and returning such a list.

`devShell.inputsFrom` is a list of packages whose deps should be in the shell. It can optionally be a function taking the package set and returning such a list.

`devShell.shellHook` is a string that provides bash code to run in shell initialization. It can optionally be a function taking the package set and returning such a string.

`devShell.env` is for setting environment variables in the shell. It is an attribute set mapping variables to values. It can optionally be a function taking the package set and returning such an attribute set.

`devShell.stdenv` is the stdenv package used for the shell. It can optionally be a function takeing the package set and returning the stdenv to use.

For example, these can be configured as follows:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      devShell = pkgs: {
        # Include build deps of emacs
        inputsFrom = [ pkgs.emacs ];
        # Add coreutils to the shell
        packages = [ pkgs.coreutils ];
        # Add shell hook. Can be a function if you need packages
        shellHook = ''
          echo Welcome to example shell!
        '';
        # Set an environment var. `env` can be an be a function
        env.TEST_VAR = "test value";
        stdenv = pkgs.clangStdenv;
      };
    };
}
```

The above exports `devShells.${system}.default` outputs.

To add the build inputs of one of your packages, you can do as follows:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      package = { stdenv }:
        stdenv.mkDerivation {
          pname = "pkg1";
          version = "0.0.1";
          src = ./;
          installPhase = "make DESTDIR=$out install";
        };
      devShell = {
        inputsFrom = pkgs: [ pkgs.pkg1 ];
      };
    };
}
}
```

To override the `devShell`, you can use a package definition as such:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      devShell = { mkShell, hello }: mkShell {
        packages = [ hello ];
      };
    };
}
}
```

devShells

Type:

```
devShells: (AttrsOf (PackageDef | Cfg | (Pkgs -> Cfg)) |
             (ModuleArgs -> (AttrsOf (PackageDef | Cfg | (Pkgs -> Cfg)))))  

Cfg.packages: [Derivation] | (Pkgs -> [Derivation])  

Cfg.inputsFrom: [Derivation] | (Pkgs -> [Derivation])  

Cfg.shellHook: Str | (Pkgs -> Str)  

Cfg.env: (AttrsOf Str) | (Pkgs -> (AttrsOf Str))  

Cfg.stdenv: Stdenv | (Pkgs -> Stdenv)
```

The `devShells` option allows you to set additional `devShell` outputs. The values each shell can be set to are the same as described above for the `devShell` option.

For example, using the configuration options:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
```

```

outputs = { flakelight, ... }:
  flakelight ./ {
    devShells.testing = {
      packages = pkgs: [ pkgs.coreutils ];
      env.TEST_VAR = "in testing shell";
    };
  };
}

```

For example, using a package definition:

```

{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      devShells.testing = { mkShell, coreutils }:
        mkShell {
          packages = [ coreutils ];
          env.TEST_VAR = "in testing shell";
        };
    };
}

```

The above flakes export `devShells.${system}.testing` outputs.

overlays

Types:

```

overlay: Overlay
overlays: (AttrsOf Overlay) | (ModuleArgs -> (AttrsOf Overlay))

```

The `overlay` and `overlays` options allow you to configure `overlays` outputs.

Multiple provided overlays for an output are merged.

The `overlay` option adds the overlay to `overlays.default`.

The `overlays` option allows you to add overlays to `overlays` outputs.

For example, to add an overlay to `overlays.default`, do the following:

```

{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      overlay = final: prev: { testValue = "hello"; };
    };
}

```

The above results in `overlays.default` output containing `testValue`.

To configure other overlays:

```
{  
    inputs.flakelight.url = "github:nix-community/flakelight";  
    outputs = { flakelight, ... }:  
        flakelight ./.  
            overlays.cool = final: prev: { testValue = "cool"; };  
    };  
}
```

The above results in a `overlays.cool` output.

checks

Types:

```
checks: (AttrsOf Check) | (Pkgs -> (AttrsOf Check))  
Check: Str | (Pkgs -> Str) | Derivation | (Pkgs -> Derivation)
```

The `checks` option allows you to add checks for `checks.${system}` attributes.

It can be set to an attribute set of checks, which can be functions, derivations, or strings.

If a check is a derivation, it will be used as is.

If a check is a string, it will be included in a bash script that runs it in a copy of the source directory, and succeeds if the no command in the string errored.

If a check is a function, it will be passed packages, and should return one of the above.

For example:

```
{  
    inputs.flakelight.url = "github:nix-community/flakelight";  
    outputs = { flakelight, ... }:  
        flakelight ./.  
            checks = {  
                # Check that succeeds if the source contains the string "hi"  
                hi = { rg, ... }: "${rg}/bin/rg hi";  
                # Check that emacs builds  
                emacs = pkgs: pkgs.emacs;  
            };  
    };  
}
```

apps

Types:

```
app: App'  
apps: (AttrsOf App') | (Pkgs -> (AttrsOf App'))
```

```
App': Str | (Pkgs -> Str) | App | (Pkgs -> App)
```

The `app` and `apps` options allow you to set `apps.${system}` outputs.

`apps` is an attribute set of apps or a function that takes packages and returns an attribute set of apps. If the app value is not an app, it is converted to a string and set as the program attr of an app. If it is a function, it is passed packages.

`app` sets `apps.default`.

For example:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      apps = {
        shell = "/bin/sh";
        emacs = pkgs: "${pkgs.emacs}/bin/emacs";
        bash = pkgs: { type = "app"; program = "${pkgs.bash}/bin/bash"; };
      };
    };
}
```

Alternatively, the above can be written as:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      apps = { emacs, bash, ... }: {
        shell = "/bin/sh";
        emacs = "${emacs}/bin/emacs";
        bash = { type = "app"; program = "${bash}/bin/bash"; };
      };
    };
}
```

templates

Types:

```
template: Template | (ModuleArgs -> Template)
templates: (AttrsOf (Template | (ModuleArgs -> Template))) |
           (ModuleArgs -> (AttrsOf (Template | (ModuleArgs -> Template))))
```

The `template` and `templates` options allow you to set `templates` outputs.

`templates` is an attribute set to template values.

`template` sets `templates.default`.

For example:

```
{  
    inputs.flakelight.url = "github:nix-community/flakelight";  
    outputs = { flakelight, ... }:  
        flakelight ./.  
            templates.test-template = {  
                path = ./test;  
                description = "test template";  
            };  
    };  
}
```

legacyPackages

Type: `Pkgs` → `Pkgs`

The `legacyPackages` option allows you to configure the flake's `legacyPackages` output. It can be set to a function that takes the package set and returns the package set to be used as the corresponding system's `legacyPackages` output.

For example:

```
{  
    inputs = {  
        flakelight.url = "github:nix-community/flakelight";  
        nixpkgs.url = "nixpkgs/nixpkgs-unstable";  
    };  
    outputs = { flakelight, nixpkgs, ... }:  
        flakelight ./.  
            legacyPackages = pkgs: nixpkgs.legacyPackages.${pkgs.system};  
    };  
}
```

To export the package set used for calling package definitions and other options that take functions passed the package set, you can do the following:

```
{  
    inputs.flakelight.url = "github:nix-community/flakelight";  
    outputs = { flakelight, ... }:  
        flakelight ./.  
            legacyPackages = pkgs: pkgs;  
    };  
}
```

formatter

Type: `Pkgs` → `Derivation`

The `formatter` option allows you to set `formatter.${system}` outputs. It can be set to a function that takes packages and returns the package to use. This overrides the `formatters` functionality described below though, so for configuring formatters for a file type, you likely want to use `formatters` instead.

For example, to use a custom formatting command:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      formatter = pkgs: pkgs.writeShellScriptBin "format-script" ''
        # Perform formatting (`nix fmt` calls the script with `.` as arg)
      '';
    };
}
```

formatters

Type: `(AttrsOf Str) | (Pkgs -> (AttrsOf Str))`

The `formatters` option allows you to configure formatting tools that will be used by `nix fmt`. If formatters are set, Flakelight will export `formatter.${system}` outputs which apply all the configured formatters.

By default, `nix` files are formatted with `nixpkgs-fmt` and `md`, `json`, and `yml` files are formatted with `prettier`.

To disable default formatters, set the `flakelight.builtinFormatters` option to false.

You can set `formatters` to an attribute set, for which the keys are a file name pattern and the value is the corresponding formatting command. `formatters` can optionally be a function that takes packages and returns the above.

Formatting tools should be added to `devShell.packages` and all packages in `devShell.packages` will be available for formatting commands.

For example, to set Rust and Zig formatters:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      devShell.packages = pkgs: [ pkgs.rustfmt pkgs.zig ];
      formatters = {
        "*.rs" = "rustfmt";
        "*.zig" = "zig fmt";
      };
}
```

```
    };
```

bundlers

Types:

```
bundler: Bundler | (Pkgs -> Bundler)
bundlers: (AttrsOf (Bundler | (Pkgs -> Bundler))) |  
           (ModuleArgs -> (AttrsOf (Bundler | (Pkgs -> Bundler))))
```

The `bundler` and `bundlers` options allow you to set `bundlers.${system}` outputs.

Each bundler value can be either a bundler function or a function that takes the package set and returns a bundler function.

`bundlers` is an attribute set of bundler values or a function that takes packages and returns an attribute set of bundler values.

`bundler` sets `bundlers.default`.

For example, a bundler that returns the passed package:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      bundler = x: x;
    };
}
```

As another example, a bundler that always returns `hello`:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      bundlers = { hello, ... }: {
        hello = x: hello;
      };
    };
}
```

To write the above using autoloads, can use the following:

```
# nix/bundlers/hello.nix
{ hello, ... }: x: hello;
```

nixosConfigurations

Type: (AttrsOf (NixOSArgs | NixOSConfig |

```

        (ModuleArgs -> (NixOSArgs | NixOSConfig))) |  

        (ModuleArgs -> (AttrsOf (NixOSArgs | NixOSConfig |  

            (ModuleArgs -> (NixOSArgs | NixOSConfig))))))

```

The `nixosConfigurations` attribute lets you set outputs for NixOS systems and home-manager users.

It should be set to an attribute set. Each value should be a set of `nixpkgs.lib.nixosSystem` args, the result of calling `nixpkgs.lib.nixosSystem`, or a function that takes `moduleArgs` and returns one of the prior.

When using a set of `nixpkgs.lib.nixosSystem` args, NixOS modules will have access to a `flake` module arg equivalent to `moduleArgs` plus `inputs'` and `outputs'`. Flakelight's `pkgs` attributes, `withOverlays`, and `packages` will also be available in the NixOS instance's `pkgs`.

When using the result of calling `nixpkgs.lib.nixosSystem`, the `config.propogationModule` value can be used as a NixOS module to gain the above benefits.

For example:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ ({ lib, config, ... }): {
      nixosConfigurations.test-system = {
        system = "x86_64-linux";
        modules = [{ system.stateVersion = "24.05"; }];
      };
    });
}
```

homeConfigurations

```

Type: (AttrsOf (HomeArgs | HomeConfig |  

    (ModuleArgs -> (HomeArgs | HomeConfig)))) |  

    (ModuleArgs -> (AttrsOf (HomeArgs | HomeConfig |  

        (ModuleArgs -> (HomeArgs | HomeConfig))))))

```

The `homeConfigurations` attribute lets you set outputs for NixOS systems and home-manager users.

It should be set to an attribute set. Each value should be a set of `home-manager.lib.homeManagerConfiguration` args, the result of calling `home-manager.lib.homeManagerConfiguration`, or a function that takes `moduleArgs` and returns one of the prior.

When using a set of `homeManagerConfiguration` args, it is required to include `system` (`pkgs` does not need to be included), and `inputs.home-manager` must

be set. home-manager modules will have access to a `flake` module arg equivalent to `moduleArgs` plus `inputs'` and `outputs'`. Flakelight's `pkgs` attributes, `withOverlays`, and `packages` will also be available in the home-manager instance's `pkgs`.

When using the result of calling `homeManagerConfiguration`, the `config.propogationModule` value can be used as a home-manager module to gain the above benefits.

For example:

```
{
  inputs = {
    flakelight.url = "github:nix-community/flakelight";
    home-manger.url = "github:nix-community/home-manager";
  };
  outputs = { flakelight, home-manager, ... }@inputs:
    flakelight ./ (config, ... ): {
      inherit inputs;
      homeConfigurations.username = {
        system = "x86_64-linux";
        modules = [{ home.stateVersion = "24.05"; }];
      };
    });
}
```

nixosModules, homeModules, and flakelightModules

Types:

```
nixosModule: Module
nixosModules: (AttrsOf Module) | (ModuleArgs -> (AttrsOf Module))
homeModule: Module
homeModules: (AttrsOf Module) | (ModuleArgs -> (AttrsOf Module))
flakelightModule: Module
flakelightModules: (AttrsOf Module) | (ModuleArgs -> (AttrsOf Module))
```

The `nixosModules`, `homeModules`, and `flakelightModules` options allow you to configure the corresponding outputs.

The `nixosModule`, `homeModule`, and `flakelightModule` options set the `default` attribute of the corresponding above option.

For example:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ (lib, ... ): {
      nixosModule = { system, lib, pkgs, ... }: {
        # nixos module configuration
      }
    }
}
```

```
    };
  });
}
```

These can be paths, which is preferred as it results in better debug output:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ ({ lib, ... }): {
      nixosModule = ./module.nix;
      homeModules = {
        default = ./home.nix;
        emacs = ./emacs.nix;
      }
    );
}
```

lib

Type: AttrSet | (ModuleArgs -> AttrSet)

The `lib` option allows you to configure the flake's `lib` output.

For example:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      lib = {
        addFive = x: x + 5;
        addFour = x: x + 4;
      };
    };
}
```

functor

Type: Outputs -> Any -> Any

The `functor` option allows you to make your flake callable.

If it is set to a function, that function will be set as the `__functor` attribute of your flake outputs.

Flakelight uses it so that calling your `flakelight` input calls `flakelight.lib.mkFlake`.

As an example:

```
{
  inputs.flakelight.url = "github:nix-community/flakelight";
  outputs = { flakelight, ... }:
    flakelight ./ {
      outputs.testvalue = 5;
      functor = self: x: x + self.testvalue;
    }
}
```

With the above flake, another flake that has imports it with the name `addFive` would be able to call `addFive 4` to get 9.

meta

Types:

```
  description: Str
  license: Str | [Str]
```

The following options are available for configuring the meta attributes of the default package for supported modules (such as `flakelight-rust` or `flakelight-zig`) or for use in your own packages through the `defaultMeta` pkgs value.

`description` allows setting the package description. By default it uses the flake description, if found.

`license` lets you set the license or license. It may be a single string or list of strings. These strings may be Spdx license identifiers or Nixpkgs license attribute names.

flakelight

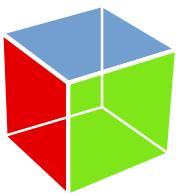
Types:

```
  flakelight.editorconfig: Bool
  flakelight.builtinFormatters: Bool
```

This option has options for configuring Flakelight's defaults.

`flakelight.editorconfig` can be set to false to disable the editorconfig check that is added if editorconfig configuration is detected.

`flakelight.builtinFormatters` can be set to false to disable the default formatting configuration.



Click, or press 's' to search

GTK

API Version: 3.0

Library Version: 3.24

Generated by

gi-docgen

2023.4

CSS Overview

Overview of CSS in GTK

This chapter describes in detail how GTK uses CSS for styling and layout.

We loosely follow the CSS value definition specification in the formatting of syntax productions.

- Nonterminals are enclosed in angle brackets (< >), all other strings that are not listed here are literals
- Juxtaposition means all components must occur, in the given order
- A double ampersand (&&) means all components must occur, in any order
- A double bar (||) means one or more of the components must occur, in any order
- A single bar (|) indicates an alternative; exactly one of the components must occur
- Brackets ([]) are used for grouping
- A question mark (?) means that the preceding component is optional
- An asterisk (*) means zero or more copies of the preceding component
- A plus (+) means one or more copies of the

CONTENT

[CSS nodes](#)
[Style sheets](#)
[Importing style sheets](#)
[Selectors](#)
[Colors](#)
[Images](#)
[Transitions](#)
[Animations](#)
[Key bindings](#)



preceding component

- A number in curly braces
({n}) means that the preceding component occurs exactly n times
- Two numbers in curly braces ({m,n}) mean that the preceding component occurs at least m times and at most n times

CSS nodes

GTK applies the style information found in style sheets by matching the selectors against a tree of nodes. Each node in the tree has a name, a state and possibly style classes. The children of each node are linearly ordered.

Every widget has one or more of these CSS nodes, and determines their name, state, style classes and how they are layed out as children and siblings in the overall node tree. The documentation for each widget explains what CSS nodes it has.

The CSS nodes of a GtkScale

```
scale[.fine-tune]  Copy
  └── marks.top
      └── mark
          └── mark
  └── trough
      └── slider
          └── [highlight]
              └── [fill]
  └── marks.bottom
      └── mark
          └── mark
```



Style sheets

The basic structure of the style sheets understood by GTK is a series of statements, which are either rule sets or "@-rules", separated by whitespace.

A rule set consists of a selector and a declaration block, which is a series of declarations enclosed in curly braces. The declarations are separated by semicolons. Multiple selectors can share the same declaration block, by putting all the separators in front of the block, separated by commas.

A rule set with two selectors

```
button, entry {    Copy  
    color: #ff00ea;  
    font: 12px "Comic Sans"  
}
```

Importing style sheets

GTK supports the CSS `import` rule, in order to load another style sheet in addition to the currently parsed one.

The syntax for `import` rules is as follows:

```
<import rule> = @import [  
    <url> = url( <string> )
```



An example for using the import rule

```
@import url("path/to/comr
```

To learn more about the `import` rule, you can read the [Cascading module](#) of the CSS specification.

Selectors

Selectors work very similar to the way they do in CSS.

All widgets have one or more CSS nodes with element names and style classes. When style classes are used in selectors, they have to be prefixed with a period. Widget names can be used in selectors like IDs. When used in a selector, widget names must be prefixed with a `#` character.

In more complicated situations, selectors can be combined in various ways. To require that a node satisfies several conditions, combine several selectors into one by concatenating them. To only match a node when it occurs inside some other node, write the two selectors after each other, separated by whitespace. To restrict the match to direct children of the parent node, insert a `>` character between the two selectors.

Theme labels that are descendants of a window

```
window label {    Copy  
background-color: #8989  
}
```



Theme notebooks, and anything within

```
notebook {           Copy  
  background-color: #a939  
}
```

Theme combo boxes, and entries that are direct children of a notebook

```
combobox,           Copy  
notebook > entry {  
  color: @fg_color;  
  background-color: #1209  
}
```

Theme any widget within a GtkBox

```
box * {           Copy  
  font: 20px Sans;  
}
```

Theme a label named title-label

```
label#title-label {Copy  
  font: 15px Sans;  
}
```

Theme any widget named main-entry

```
#main-entry {           Copy  
  background-color: #f0a8  
}
```

Theme all widgets with the style class entry

```
.entry {           Copy  
  color: #39f1f9;  
}
```

Theme the entry of a



GtkSpinButton

```
spinbutton entry {Copy  
color: #900185;  
}
```

It is possible to select CSS nodes depending on their position amongst their siblings by applying pseudo-classes to the selector, like `:first-child`, `:last-child` or `:nth-child(even)`. When used in selectors, pseudo-classes must be prefixed with a `:` character.

Theme labels in the first notebook tab

```
notebook tab:first-child {Copy  
color: #89d012;  
}
```

Another use of pseudo-classes is to match widgets depending on their state. The available pseudo-classes for widget states are `:active`, `:hover`, `:disabled`, `:selected`, `:focus`, `:indeterminate`, `:checked` and `:backdrop`. In addition, the following pseudo-classes don't have a direct equivalent as a widget state: `:dir(ltr)` and `:dir rtl` (for text direction); `:link` and `:visited` (for links); `:drop(active)` (for highlighting drop targets). Widget state pseudo-classes may only apply to the last element in a selector.

Theme pressed buttons



```
button:active {      Copy  
background-color: #0274  
}
```

Theme buttons with the mouse pointer over it

```
button:hover {      Copy  
background-color: #3085  
}
```

Theme insensitive widgets

```
*:disabled {      Copy  
background-color: #320a  
}
```

Theme checkbuttons that are checked

```
checkbox:checked {      Copy  
background-color: #56f9  
}
```

Theme focused labels

```
label:focus {      Copy  
background-color: #b494  
}
```

Theme inconsistent checkbuttons

```
checkbox:indeterminate {      Copy  
background-color: #2039  
}
```

To determine the effective style for a widget, all the matching rule sets are merged. As in CSS, rules apply by specificity, so the rules whose selectors more closely match a node will take precedence over the others.



The full syntax for selectors understood by GTK can be found in the table below. The main difference to CSS is that GTK does not currently support attribute selectors.

| Pattern | Matches | Reference | Notes |
|-------------------------------|--|---------------------|--|
| * | any node | CSS | - |
| E | any node with name E | CSS | - |
| E.class | any E node with the given style class | CSS | - |
| E#id | any E node with the given ID | CSS | GTK uses the widget name as ID |
| E:nth-child(<nth-child>) | any E node which is the n-th child of its parent node | CSS | - |
| E:nth-last-child(<nth-child>) | any E node which is the n-th child of its parent node, counting from the end | CSS | - |
| E:first-child | any E node which is the first child of its parent node | CSS | - |
| E:last-child | any E node which is the last child of its parent node | CSS | - |
| E:only-child | any E node which is the only child of its parent node | CSS | Equivalent to E:first-child:last-child |



| Pattern | Matches | Reference | Notes |
|----------------------------|---|--|--|
| E:link, E:visited | any E node which represents a hyperlink, not yet visited (:link) or already visited (:visited) | CSS | Corresponds to GTK_STATE _FLAG_LINK and GTK_STATE _FLAGS_VISITED |
| E:active, E:hover, E:focus | any E node which is part of a widget with the corresponding state | CSS | Corresponds to GTK_STATE _FLAG_ACTIVE , GTK_STATE _FLAG_PRELIGHT and GTK_STATE _FLAGS_FOCUSUSED ; GTK also allows E:prelight and E:focused |
| E:disabled | any E node which is part of a widget which is disabled | CSS | Corresponds to GTK_STATE _FLAG_INSENSITIVE ; GTK also allows E:insensitive |
| E:checked | any E node which is part of a widget (e.g. radio- or checkbuttons) which is checked | CSS | Corresponds to GTK_STATE _FLAG_CHECKED |
| E:indeterminate | any E node which is part of a widget (e.g. radio- or checkbuttons) which is in an indeterminate state | CSS3 , CSS4 | Corresponds to GTK_STATE _FLAG_INCONSISTENT ; GTK also allows E:inconsistent |

| Pattern | Matches | Reference | Notes |
|---------------------------|---|-----------|---|
| E:backdrop, E:selected | any E node which is part of a widget with the corresponding state | - | Corresponds to GTK_STATE _FLAG_BACK DROP , GTK_STATE _FLAG_SELE CTED |
| E:not(<selector>) | any E node which does not match the simple selector <selector> | CSS | - |
| E:dir(ltr), E:dir(rtl) | any E node that has the corresponding text direction | CSS4 | - |
| E:drop(active) | any E node that is an active drop target for a current DND operation | CSS4 | - |
| E F | any F node which is a descendent of an E node | CSS | - |
| E > F | any F node which is a child of an E node | CSS | - |
| E ~ F | any F node which is preceded by an E node | CSS | - |
| E + F | any F node which is immediately preceded by an E node | CSS | - |

Where:

<nth-child> = even  odd



To learn more about selectors in CSS, read the [Selectors module](#) of the CSS specification.

Colors

CSS allows to specify colors in various ways, using numeric values or names from a predefined list of colors.

```
<color> = currentColor |  
<rgb color> = rgb( <number>,  
<rgba color> = rgba( <number>,  
<hex color> = #<hex digit>  
<alpha value> = <number> ,
```

The keyword `currentColor` resolves to the current value of the color property when used in another property, and to the inherited value of the color property when used in the color property itself.

The keyword `transparent` can be considered a shorthand for `rgba(0,0,0,0)`.

For a list of valid color names and for more background on colors in CSS, see the [Color module](#) of the CSS specification.

Specifying colors in various ways

```
color: transparent;  
background-color: red;  
border-top-color: rgb(128  
border-left-color: rgba(1  
border-right-color: #ff00  
border-bottom-color: #fff
```

GTK adds several additional ways to specify colors.



```
<gtk color> = <symbolic copy
```

The first is a reference to a color defined via a `define`-`color` rule. The syntax for define -color rules is as follows:

```
<define color rule>opy@de
```

To refer to the color defined by a `define -color rule`, use the name from the rule, prefixed with `@`.

```
<symbolic color> = @<nameopy
```

An example for defining colors

```
@define-color bg_color #fopy
*
* {
    background-color: @bg_c
}
```

GTK also supports color expressions, which allow colors to be transformed to new ones and can be nested, providing a rich language to define colors. Color expressions resemble functions, taking 1 or more colors and in some cases a number as arguments.

`shade()` leaves the color unchanged when the number is 1 and transforms it to black or white as the number approaches 0 or 2 respectively. For `mix()`, 0 or 1 return the unaltered 1st or 2nd color respectively; numbers between 0 and 1 return blends of the two; and numbers



below 0 or above 1 intensify the RGB components of the 1st or 2nd color respectively.
alpha() takes a number from 0 to 1 and applies that as the opacity of the supplied color.

`<color expression> Copy high alph`

On Windows, GTK allows to refer to system colors, as follows:

`<win32 color> = -gtk-win3 Copy`

Images

CSS allows to specify images in various ways, for backgrounds and borders.

`<image> = <url> | Copy
<crossfade> = cross-fade(
<alternatives> = image([
<gradient> = <linear grad
<linear gradient> = [lin
[[
<cc
<radial gradient> = [rad
[[
<cc
<side or corner> = [left
<color stops> = <color s
<color stop> = <color> [
<shape> = circle | ellips
<size> = <extent keyword>
<extent keyword> = closes`

The simplest way to specify an image in CSS is to load an image file from a URL. CSS does not specify anything about supported file formats; within GTK, you can expect at least PNG, JPEG and SVG to work. The full list of supported image formats is determined by the available gdk-pixbuf image loaders and may vary



between systems.

Loading an image file

```
button {  
    background-image: url("Copy")  
}
```

A crossfade lets you specify an image as an intermediate between two images.

Crossfades are specified in the draft of the level 4 [Image module](#) of the CSS specification.

Crossfading two images

```
button {  
    background-image: cross(Copy)  
}
```

The `image()` syntax provides a way to specify fallbacks in case an image format may not be supported. Multiple fallback images can be specified, and will be tried in turn until one can be loaded successfully. The last fallback may be a color, which will be rendered as a solid color image.

Image fallback

```
button {  
    background-image: image(Copy)  
}
```



Gradients are images that smoothly fades from one color to another. CSS provides ways to specify repeating and non-repeating linear and radial gradients. Radial

gradients can be circular, or axis-aligned ellipses. In addition to CSS gradients, GTK has its own `-gtk-gradient` extensions.

A linear gradient is created by specifying a gradient line and then several colors placed along that line. The gradient line may be specified using an angle, or by using direction keywords.

Linear gradients

```
button {  
    background-image: linear-gradient(  
        top left,  
        color-stop(0%, red),  
        color-stop(50%, yellow),  
        color-stop(100%, blue)  
    );  
}  
  
label {  
    background-image: linear-gradient(  
        to right,  
        color-stop(0%, red),  
        color-stop(50%, yellow),  
        color-stop(100%, blue)  
    );  
}
```

A radial gradient is created by specifying a center point and one or two radii. The radii may be given explicitly as lengths or percentages or indirectly, by keywords that specify how the end circle or ellipsis should be positioned relative to the area it is drawn in.

Radial gradients

```
button {  
    background-image: radial-gradient(  
        center,  
        color-stop(0%, red),  
        color-stop(50%, yellow),  
        color-stop(100%, blue)  
    );  
}  
  
label {  
    background-image: radial-gradient(  
        center,  
        color-stop(0%, red),  
        color-stop(50%, yellow),  
        color-stop(100%, blue)  
    );  
}
```

To learn more about gradients in CSS, including details of how color stops are placed on the gradient line and keywords for specifying radial sizes, you can read the [Image module](#) of



the CSS specification.

GTK extends the CSS syntax for images and also uses it for specifying icons.

`<gtk image> = <gtk gradient>`

GTK supports an alternative syntax for linear and radial gradients (which was implemented before CSS gradients were supported).

`<gtk gradient> = <gtk linear gradient> = -`

`<gtk radial gradient> = -`

`<x position> = left | right
<y position> = top | bottom
<radius> = <number>
<gtk color stops> = <gtk color stop>
<gtk color stop> = color-`

The numbers used to specify x and y positions, radii, as well as the positions of color stops, must be between 0 and 1. The keywords for x and y positions (left, right, top, bottom, center), map to numeric values of 0, 1 and 0.5 in the obvious way. Color stops using the from() and to() syntax are abbreviations for color-stop with numeric positions of 0 and 1, respectively.

Linear gradients

`button {
background-image: -gtk-`

`}
label {
background-image: -gtk-`



}

Radial gradients

```
button {           Copy  
background-image: -gtk-
```

```
}
```

```
label {           Copy  
background-image: -gtk-
```

}

GTK has extensive support for loading icons from icon themes. It is accessible from CSS with the `-gtk-icontheme` syntax.

```
<themed icon> = -gtkicon
```

The specified icon name is used to look up a themed icon, while taking into account the values of the `-gtk-icon-theme` and `-gtk-icon-palette` properties. This kind of image is mainly used as value of the `-gtk-icon-source` property.

Using themed icons in CSS

```
spinner {           Copy  
-gtk-icon-source: -gtk-  
-gtk-icon-palette: succ  
}  
arrow.fancy {           Copy  
-gtk-icon-source: -gtk-  
-gtk-icon-theme: 'Oxyge  
}
```



GTK supports scaled rendering on hi-resolution displays. This works best if images can specify normal and hi-resolution variants. From CSS, this can be done with the `-gtk-scaled` syntax.

```
<scaled image> = -Copy gtk-sca
```

While `-gtk-scaled` accepts multiple higher-resolution variants, in practice, it will mostly be used to specify a regular image and one variant for scale 2.

Scaled images in CSS

```
arrow { Copy  
-gtk-icon-source: -gtk-  
}
```

```
<recolored image> = Copy gtk-
```

Symbolic icons from the icon theme are recolored according to the `-gtk-icon-palette` property. The recoloring is sometimes needed for images that are not part of an icon theme, and the `-gtk-recolor` syntax makes this available. `-gtk-recolor` requires a url as first argument. The remaining arguments specify the color palette to use. If the palette is not explicitly specified, the current value of the `-gtk-icon-palette` property is used.



Recoloring an image

```
arrow { Copy
```

```
-gtk-icon-source: -gtk-  
}
```

On Windows, GTK allows to refer to system theme parts as images, as follows:

(win32 theme part) [Copy](#) gtk

Transitions

CSS defines a mechanism by which changes in CSS property values can be made to take effect gradually, instead of all at once. GTK supports these transitions as well.

To enable a transition for a property when a rule set takes effect, it needs to be listed in the transition-property property in that rule set. Only animatable properties can be listed in the transition-property.

The details of a transition can be modified with the transition-duration, transition-timing-function and transition-delay properties.

To learn more about transitions, you can read the [Transitions module](#) of the CSS specification.

Animations

In addition to transitions, which are triggered by changes of the underlying node tree, CSS also supports defined animations. While transitions specify how



property values change from one value to a new value, animations explicitly define intermediate property values in keyframes.

Keyframes are defined with an @-rule which contains one or more of rule sets with special selectors. Property declarations for nonanimatable properties are ignored in these rule sets (with the exception of animation properties).

```
<keyframe rule> = Copy
<animation rule> = <anima
<animation selector> = <s
<single animation selecto
```

To enable an animation, the name of the keyframes must be set as the value of the animation-name property. The details of the animation can be modified with the animation-duration, animation-timing-function, animation-iteration-count and other animation properties.

A CSS animation

```
@keyframes spin { Copy
  to { -gtk-icon-transfor
}
spinner {
  animation-name: spin;
  animation-duration: 1s;
  animation-timing-functi
  animation-iteration-cou
}
```

To learn more about animations, you can read the [Animations module](#) of the CSS specification.



Key bindings

In order to extend key bindings affecting different widgets, GTK supports the `binding-set` rule to parse a set of `bind` / `unbind` directives. Note that in order to take effect, the binding sets defined in this way must be associated with rule sets by setting the `-gtk-key-bindings` property.

The syntax for `binding-set` rules is as follows:

```
<binding set rule> <Copy>
<binding> = bind "<accelerator>" <signal emission> = "<signal name>"
<unbinding> = unbind "<accelerator>" <argument> <Copy>
```

where `<accelerator>` is a string that can be parsed by `gtk_accelerator_parse()`, `<signal name>` is the name of a keybinding signal of the widget in question, and the `<argument>` list must be according to the signals declaration.

An example for using the binding-set rule

```
@binding-set binding-set1 <Copy>
  bind "<alt>Left" { "move-left";
  unbind "End";
};

@binding-set binding-set2 <Copy>
  bind "<alt>Right" { "move-right";
  bind "<alt>KP_Space" {
};

entry {
  -gtk-key-bindings: bind
}
```



#+TITLE: Impermanence

Lets you choose what files and directories you want to keep between reboots - the rest are thrown away.

Why would you want this?

- It keeps your system clean by default.
- It forces you to declare settings you want to keep.
- It lets you experiment with new software without cluttering up your system.

There are a few different things to set up for this to work:

- A root filesystem which somehow gets wiped on reboot. There are a few ways to achieve this. See the [[#system-setup][System setup]] section for more info.
- At least one mounted volume where the files and directories you want to keep are stored permanently.
- At least one of the modules in this repository, which take care of linking or bind mounting files between the persistent storage mount point and the root file system. See the [[#module-usage][Module usage]] section for more info.
- Contact
Join the [[https://matrix.to/#/#impermanence:nixos.org][matrix room]] to chat about the project.
- System setup

There are many ways to wipe your root partition between boots. This section lists a few common ways to accomplish this, but is by no means an exhaustive list.

*** tmpfs

The easiest method is to use a tmpfs filesystem for the root. This is the easiest way to set up impermanence on systems which currently use a traditional filesystem (ext4, xfs, etc) as the root filesystem, since you don't have to repartition.

All data stored in tmpfs only resides in system memory, not on disk. This automatically takes care of cleaning up between boots, but also comes with some pretty significant drawbacks:

- Downloading big files or trying programs that generate large amounts of data can easily result in either an out-of-memory or disk-full scenario.

- If the system crashes or loses power before you've had a chance to move files you want to keep to persistent storage, they're gone forever.

Using tmpfs as the root filesystem, the filesystem setup would look something like this:

```
#+begin_src nix
{
  fileSystems."/" = {
    device = "none";
    fsType = "tmpfs";
    options = [ "defaults" "size=25%" "mode=755" ];
  };

  fileSystems."/persistent" = {
    device = "/dev/root_vg/root";
    neededForBoot = true;
    fsType = "btrfs";
    options = [ "subvol=persistent" ];
  };

  fileSystems."/nix" = {
    device = "/dev/root_vg/root";
    fsType = "btrfs";
    options = [ "subvol=nix" ];
  };

  fileSystems."/boot" = {
    device = "/dev/disk/by-uuid/XXXX-XXXX";
    fsType = "vfat";
  };
}
#+end_src
```

where the `size` option determines how much system memory is allowed to be used by the filesystem.

*** BTRFS subvolumes

A more advanced solution which doesn't have the same drawbacks as using tmpfs is to use a regular filesystem, but clean it up between boots. A relatively easy way to do this is to use BTRFS and create a new subvolume to use as root on boot. This also allows you to keep a number of old roots around, in case of crashes, power outages or other accidents.

A setup which would remove automatically remove roots that are older than 30 days could look like this:

```
#+begin_src nix
{
    fileSystems."/" = {
        device = "/dev/root_vg/root";
        fsType = "btrfs";
        options = [ "subvol=root" ];
    };

    boot.initrd.postDeviceCommands = lib.mkAfter ''
        mkdir /btrfs_tmp
        mount /dev/root_vg/root /btrfs_tmp
        if [[ -e /btrfs_tmp/root ]]; then
            mkdir -p /btrfs_tmp/old_roots
            timestamp=$(date --date="@$(stat -c %Y /btrfs_tmp/root)" "+%Y-%m--%d_%H:%M:%S")
            mv /btrfs_tmp/root "/btrfs_tmp/old_roots/$timestamp"
        fi

        delete_subvolume_recursively() {
            IFS=$'\n'
            for i in $(btrfs subvolume list -o "$1" | cut -f 9- -d ' '); do
                delete_subvolume_recursively "/btrfs_tmp/$i"
            done
            btrfs subvolume delete "$1"
        }

        for i in $(find /btrfs_tmp/old_roots/ -maxdepth 1 -mtime +30); do
            delete_subvolume_recursively "$i"
        done

        btrfs subvolume create /btrfs_tmp/root
        umount /btrfs_tmp
    '';

    fileSystems."/persistent" = {
        device = "/dev/root_vg/root";
        neededForBoot = true;
        fsType = "btrfs";
        options = [ "subvol=persistent" ];
    };

    fileSystems."/nix" = {
        device = "/dev/root_vg/root";
    };
}
```

```

        fsType = "btrfs";
        options = [ "subvol=nix" ];
    };

    fileSystems."/boot" = {
        device = "/dev/disk/by-uuid/XXXX-XXXX";
        fsType = "vfat";
    };
}
#+end_src

```

This assumes the BTRFS filesystem can be found in an LVM volume group called `~root_vg~`. Adjust the path as necessary.

- Module usage

There are currently two modules: one for `NixOS` and one for `home-manager`.

*** NixOS

To use the module, import it into your configuration with

```

#+begin_src nix
{
  imports = [ /path/to/impermanence/nixos.nix ];
}
#+end_src

```

or use the provided `~nixosModules.impermanence~` flake output:

```

#+begin_src nix
{
  inputs = {
    impermanence.url = "github:nix-community/impermanence";
  };

  outputs = { self, nixpkgs, impermanence, ... }:
  {
    nixosConfigurations.sythe = nixpkgs.lib.nixosSystem {
      system = "x86_64-linux";
      modules = [
        impermanence.nixosModules.impermanence
        ./machines/sythe/configuration.nix
      ];
    };
  };
}
#+end_src

```

This adds the `~environment.persistence~` option, which is an attribute set of submodules, where the attribute name is the path to persistent storage.

Usage is shown best with an example:

```
#+begin_src nix
{
  environment.persistence."/persistent" = {
    hideMounts = true;
    directories = [
      "/var/log"
      "/var/lib/bluetooth"
      "/var/lib/nixos"
      "/var/lib/systemd/coredump"
      "/etc/NetworkManager/system-connections"
      { directory = "/var/lib/colord"; user = "colord"; group = "colord"; mode = "u=rwx,g=,o="; };
    ];
    files = [
      "/etc/machine-id"
      { file = "/var/keys/secret_file"; parentDirectory = { mode = "u=rwx,g=,o="; }; };
    ];
    users.talyz = {
      directories = [
        "Downloads"
        "Music"
        "Pictures"
        "Documents"
        "Videos"
        "VirtualBox VMs"
        { directory = ".gnupg"; mode = "0700"; }
        { directory = ".ssh"; mode = "0700"; }
        { directory = ".nixops"; mode = "0700"; }
        { directory = ".local/share/keyrings"; mode = "0700"; }
        ".local/share/direnv"
      ];
      files = [
        ".screenrc"
      ];
    };
  };
}
#+end_src

- ~/persistent is the path to your persistent storage location
```

This allows for multiple different persistent storage locations. If you, for example, have one location you back up and one you don't, you can use both by defining two separate attributes under `~environment.persistence~`.

- `~directories~` are all directories you want to bind mount to persistent storage. A directory can be represented either as a string, simply denoting its path, or as a submodule. The submodule representation is useful when the default assumptions, mainly regarding permissions, are incorrect. The available options are:
 - `~directory~`, the path to the directory you want to bind mount to persistent storage. Only setting this option is equivalent to the string representation.
 - `~persistentStoragePath~`, the path to persistent storage. Defaults to the `~environment.persistence~` submodule name, i.e. `~"/persistent"~` in the example. This should most likely be left to its default value - don't change it unless you're certain you really need to.
 - `~user~`, the user who should own the directory. If the directory doesn't already exist in persistent storage, it will be created and this user will be its owner. This also applies to any parent directories which don't yet exist. Changing this once the directory has been created has no effect.
 - `~group~`, the group who should own the directory. If the directory doesn't already exist in persistent storage, it will be created and this group will be its owner. This also applies to any parent directories which don't yet exist. Changing this once the directory has been created has no effect.
 - `~mode~`, the permissions to set for the directory. If the directory doesn't already exist in persistent storage, it will be created with this mode. Can be either an octal mode (e.g. `~0700~`) or a symbolic mode (e.g. `~u=rwx,g=,o=~`). Parent directories that don't yet exist are created with default permissions. Changing this once the directory has been created has no effect.
- `~files~` are all files you want to link or bind to persistent storage. A file can be represented either as a string, simply denoting its path, or as a submodule. The submodule

representation is useful when the default assumptions, mainly regarding the permissions of its parent directory, are incorrect. The available options are:

- `~file~`, the path to the file you want to bind mount to persistent storage. Only setting this option is equivalent to the string representation.
- `~persistentStoragePath~`, the path to persistent storage. Defaults to the `~environment.persistence~` submodule name, i.e. `"/persistent"` in the example. This should most likely be left to its default value - don't change it unless you're certain you really need to.
- `~parentDirectory~`, the permissions that should be applied to the file's parent directory, if it doesn't already exist. Available options are `~user~`, `~group~` and `~mode~`. See their definition in `~directories~` above.

If the file exists in persistent storage, it will be bind mounted to the target path; otherwise it will be symlinked.

- `~hideMounts~` allows you to specify whether to hide the bind mounts from showing up as mounted drives in the file manager. If enabled, it sets the mount option `~x-gvfs-hide~` on all the bind mounts.
- `~users.talyz~` handles files and directories in `~talyz~'s home directory`

The `~users~` option defines a set of submodules which correspond to the users' names. The `~directories~` and `~files~` options of each submodule work like their root counterparts, but the paths are automatically prefixed with with the user's home directory.

If the user has a non-standard home directory (i.e. not `~/home/<username>~`), the `~users.<username>.home~` option has to be set to this path - it can't currently be automatically deduced due to a limitation in `nixpkgs~`.

/Important note:/ Make sure your persistent volumes are marked with `~neededForBoot~`, otherwise you will run into problems.

*** home-manager

Usage of the `~home-manager~` module is very similar to the one of the `NixOS~` module - the key differences are that the `~persistence~` option

is now under `~home~`, rather than `~environment~`, and the addition of the submodule option `~removePrefixDirectory~`.

/Important note:/ You have to use the `~home-manager~` `~NixOS~` module (in the `~nixos~` directory of `~home-manager~`'s repo) in order for this module to work as intended.

To use the module, import it into your configuration with

```
#+begin_src nix
{
  imports = [ /path/to/impermanence/home-manager.nix ];
}
#+end_src
```

This adds the `~home.persistence~` option, which is an attribute set of submodules, where the attribute name is the path to persistent storage.

Usage is shown best with an example:

```
#+begin_src nix
{
  home.persistence."/persistent/home/talzy" = {
    directories = [
      "Downloads"
      "Music"
      "Pictures"
      "Documents"
      "Videos"
      "VirtualBox VMs"
      ".gnupg"
      ".ssh"
      ".nixops"
      ".local/share/keyrings"
      ".local/share/direnv"
    {
      directory = ".local/share/Steam";
      method = "symlink";
    }
  ];
  files = [
    ".screenrc"
  ];
  allowOther = true;
};
```

```

        }
#+end_src

- ~/persistent/home/talzyz is the path to your persistent storage location
- ~directories~ are all directories you want to link to persistent storage
  - It is possible to switch the linking ~method~ between bindfs (the
    default) and symbolic links.
- ~files~ are all files you want to link to persistent storage. These are
  symbolic links to their target location.
- ~allowOther~ allows other users, such as ~root~, to access files
  through the bind mounted directories listed in
  ~directories~. Useful for ~sudo~ operations, Docker, etc. Requires
  the NixOS configuration ~programs.fuse.userAllowOther = true~.
```

Additionally, the `~home-manager~` module allows for compatibility with `~dotfiles~` repos structured for use with [\[\[https://www.gnu.org/software/stow/\]\]](https://www.gnu.org/software/stow/) [GNU Stow]. Files linked to are one level deeper than where they should end up. This can be achieved by setting `~removePrefixDirectory~` to `~true~`:

```

#+begin_src nix
{
  home.persistence."/etc/nixos/home-talzyz-nixpkgs/dotfiles" = {
    removePrefixDirectory = true;
    files = [
      "screen/.screenrc"
    ];
    directories = [
      "fish/.config/fish"
    ];
  };
}
#+end_src
```

In the example, the `~.screenrc~` file and `~.config/fish~` directory should be linked to from the home directory; `~removePrefixDirectory~` removes the first part of the path when deciding where to put the links.

`/Note:/` When using `~bindfs~` fuse filesystem for directories, the names of the directories you add will be visible in the `~/etc/mtab~` file and in the output of `~mount~` to all users.

** Further reading The following blog posts provide more information on the concept of ephemeral roots:

- <https://elis.nu/blog/2020/05/nixos-tmpfs-as-root/> — [[https://github.com/etu/][@etu]]’s blog post walks the reader through a NixOS-on-tmpfs installation.

- <https://grahamc.com/blog/erase-your-darlings> — [[<https://github.com/grahamc/>][@grahamc]]’s blog post details why one would want to erase their state at every boot, as well as how to achieve this using ZFS snapshots.

GJS

Intro

[Platform and API](#)[GNOME Platform](#)[Web APIs](#)[Imports and Modules](#)[ES Modules](#)[Legacy Imports](#)[Classes](#)[ES Classes](#)[GObject Classes](#)[Legacy Classes](#)[Asynchronous Programming](#)[Style Guide](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)

GNOME JavaScript Introduction

Welcome to GNOME JavaScript (GJS)! This introduction covers the basics of how JavaScript is used with GJS. More detailed information about GJS is available in the [GJS Usage Documentation](#).

Platform and API

Many APIs that JavaScript developers are familiar with are actually a part of the [Web API](#), such as DOM or Fetch. GJS is not intended as an environment for web development and does not include support for most of these APIs.

GNOME Platform

TIP

GNOME Platform APIs are documented at [gjs-docs.gnome.org](https://docs.gnome.org/gjs/)

GJS provides JavaScript bindings for the GNOME platform libraries, meaning that developers will use libraries like [Gio](#) for working with files, [Gtk](#) to build user interfaces, [Soup](#) for WebSockets and so on.



GJS

Intro

[Platform and API](#)[GNOME Platform](#)[Web APIs](#)[Imports and Modules](#)[ES Modules](#)[Legacy Imports](#)[Classes](#)[ES Classes](#)[GObject Classes](#)[Legacy Classes](#)[Asynchronous Programming](#)[Style Guide](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)

Web APIs

TIP

See the documentation on [GJS Usage](#) for details about built-in modules and APIs.

Since the adoption of ES Modules, GJS has gained support for some widely used standards from the [Web API](#). These APIs may be more familiar to developers that have used other JavaScript environments.

Console API

As of GJS 1.70 (GNOME 41), the Console API is available as described in the [WHATWG Console Standard](#). The `console` object is available globally without import.

Encoding API

As of GJS 1.70 (GNOME 41), the Encoding API is available as described in the [WHATWG Encoding Standard](#). The `TextDecoder` and `TextEncoder` objects are available globally without import.

Timers API

As of GJS 1.70 (GNOME 41), the Timers API is available as described in the [WHATWG HTML Standard](#). The `setTimeout()`, `setInterval()`, `clearTimeout()` and `clearInterval()` methods are available globally without import.



GJS

Intro

- [Platform and API](#)
- [GNOME Platform](#)
- [Web APIs](#)
- [Imports and Modules](#)
- [ES Modules](#)
- [Legacy Imports](#)
- [Classes](#)
- [ES Classes](#)
- [GObject Classes](#)
- [Legacy Classes](#)

Asynchronous Programming

- [Style Guide](#)
- [Tips On Memory Management](#)

GLib

GVariant

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)
- [GType](#)
- [GValue](#)
- [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)

Imports and Modules

TIP

See the [Extensions Documentation](#) for ES Module usage in GNOME Shell Extensions.

GJS uses standard ES Modules for imports and modules, but includes additional specifiers for platform libraries like `Gtk`.

ES Modules

TIP

See the [GJS Documentation](#) for details about using ES Modules in GJS.

As of GJS 1.68 (GNOME 40), standard [ES Modules](#) are supported and the preferred import method.

```
/* GJS's Built-in modules have custom specifier
   import Cairo from 'cairo';
   import Gettext from 'gettext';
   import System from 'system';
```

```
/* Platform libraries use the gi:// URI in the specifier
   import GLib from 'gi://GLib';
   import Gio from 'gi://Gio';
```

```
/* Platform libraries with multiple versions import
   import Gtk from 'gi://Gtk?version=4.0';
```

```
/* User modules may be imported using a relative specifier
   import MyModule from './my-module';
```



GJS

Intro

- [Platform and API](#)
- [GNOME Platform](#)
- [Web APIs](#)
- [Imports and Modules](#)
 - [ES Modules](#)
 - [Legacy Imports](#)
- [Classes](#)
 - [ES Classes](#)
 - [GObject Classes](#)
 - [Legacy Classes](#)
- [Asynchronous Programming](#)
- [Style Guide](#)
- [Tips On Memory Management](#)

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

D-Bus

File Operations

```
/*  
 * file extension is included in the imports  
 */  
import * as Utils from './lib/utils.js';
```

Legacy Imports

WARNING

ES Modules should be preferred for new projects and existing projects should migrate when possible.

Prior to the ECMAScript standard for modules, GJS used a custom `imports` object.

```
/* GJS's Built-in modules are top-level properties  
 * of the global object.  
 */  
const Cairo = imports.cairo;  
const Gettext = imports.gettext;  
const System = imports.system;
```

```
/* Platform libraries are properties of `gi`.  
 * They are loaded automatically.  
 */  
const GLib = imports.gi.GLib;  
const Gio = imports.gi.Gio;
```

```
/* Platform libraries with multiple versions are loaded  
 * automatically.  
 */  
imports.gi.versions.Gtk = '4.0';  
const Gtk = imports.gi.Gtk;
```

```
/* User modules may be imported using a relative path.  
 *  
 * `utils.js` is in the subdirectory `lib`, so we  
 * file extension is omitted from the import.  
 */  
const Utils = imports.lib.utils;
```

GJS

Intro

[Platform and API](#)[GNOME Platform](#)[Web APIs](#)[Imports and Modules](#)[ES Modules](#)[Legacy Imports](#)[Classes](#)[ES Classes](#)[GObject Classes](#)[Legacy Classes](#)[Asynchronous Programming](#)[Style Guide](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)

Classes

GJS uses standard JavaScript syntax for classes when possible, but includes additional syntax for GObject classes.

ES Classes

GJS uses standard [ES Classes](#) for JavaScript classes.

```
js
class ClassExample {
    constructor() {
        this._initialized = true;
    }

    get example_property() {
        if (this._example_property === undefined)
            this._example_property = null;

        return this._example_property;
    }

    set example_property(value) {
        if (this.example_property === value)
            return;

        this._example_property = value;
    }
}
```



GObject Classes

TIP

GJS

Intro

[Platform and API](#)[GNOME Platform](#)[Web APIs](#)[Imports and Modules](#)[ES Modules](#)[Legacy Imports](#)[Classes](#)[ES Classes](#)[GObject Classes](#)[Legacy Classes](#)[Asynchronous Programming](#)[Style Guide](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)

The GNOME platform is built on the GObject type system, which brings object-oriented concepts to C. GJS uses special syntax to integrate with GObject.

```
import GObject from 'gi://GObject';

const SubclassExample = GObject.registerClass
  GTypeName: 'SubclassExample',
  Properties: {
    'example-property': GObject.ParamSpec
      'example-property',
      'Example Property',
      'A read-write boolean property',
      GObject.ParamFlags.READWRITE,
      true
    },
  },
  Signals: {
    'example-signal': {},
  },
}, class SubclassExample extends GObject.Object {
  constructor(constructProperties = {}) {
    super(constructProperties);
  }

  get example_property() {
    if (this._example_property === undefined)
      this._example_property = null;

    return this._example_property;
  }

  set example_property(value) {
    if (this.example_property === value)
      return;
  }
}
```



GJS

Intro

[Platform and API](#)[GNOME Platform](#)[Web APIs](#)[Imports and Modules](#)[ES Modules](#)[Legacy Imports](#)[Classes](#)[ES Classes](#)[GObject Classes](#)[Legacy Classes](#)[Asynchronous Programming](#)[Style Guide](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)

```
});
```

Legacy Classes

WARNING

ES Classes should be preferred for new projects and existing projects should migrate when possible.

Prior to the ECMAScript standard for classes, GJS used a custom syntax to integrate with GObject.

```
const Lang = imports.lang;
```

```
const GObject = imports.gi.GObject;
```

```
var ExampleSubclass = new Lang.Class({
  Name: 'ExampleSubclass',
  GTypeName: 'ExampleSubclass',
  Signals: {},
  InternalChildren: [],
  Children: [],
  Extends: GObject.Object,
  _init(constructArguments) {
    this.parent(constructArguments);
  },
});
```



Contributors: Last Updated: 11/11/2023, 12:15:54 PM
Andy Holmes, Evan Welsh, Detlev Zundel, Marcin Jahn,

GJS**Intro**[Platform and API](#)[GNOME Platform](#)[Web APIs](#)[Imports and Modules](#)[ES Modules](#)[Legacy Imports](#)[Classes](#)[ES Classes](#)[GObject Classes](#)[Legacy Classes](#)[Asynchronous Programming](#)[Style Guide](#)[Tips On Memory Management](#)**GLib**[GVariant](#)**GObject**[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)**Gio**[Actions and Menus](#)[D-Bus](#)[File Operations](#)[Asynchronous Programming →](#)

GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

Asynchronous Programming

JavaScript is a single-threaded, concurrent programming language. Concurrency is achieved by using an event loop that halts while processing an event, before returning to process more events.

The [Promise](#) API is a powerful framework for controlling execution flow in an event loop, while keeping code simple and maintainable.

Although JavaScript can only be run in a single-thread environment, the GNOME APIs contain many functions that use per-task threads to execute blocking operations.

Using these three tools together allows GJS programmers to schedule events based on their priority, keep code clean and understandable, and indirectly use threads to write responsive and performant code.

The Main Loop

In order to process events in a concurrent fashion, a JavaScript engine needs an event loop. Although GJS is powered by Firefox's [SpiderMonkey](#) JavaScript engine, it uses GLib's [Event Loop](#). The event loop is the foundation of concurrent and asynchronous programming in GJS, so we will cover it in some detail.



GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

Gtk.Application or Adw.Application , a main loop will be started for you when you call

[Gio.Application.run\(\)](#). If you are writing a GNOME Shell Extension, you will be using the main loop already running in GNOME Shell.

It's still useful to know how to create a main loop for simple scripts, so let's get started with an example of creating a GLib.MainLoop and adding a timeout source:

```
1 import GLib from 'gi://GLib';  
2  
3 // Here we're creating an event loop, to  
4 const loop = new GLib.MainLoop(null, false);  
5  
6 // Here we're adding a timeout source to  
7 // the loop, which will trigger a  
8 // callback after one second. The return value  
9 const sourceId = GLib.timeout_add_seconds(1,  
10     GLib.PRIORITY_DEFAULT,           //  
11     1,                            //  
12     () => {                      //  
13         return GLib.SOURCE_CONTINUE; //  
14     }  
15 );  
16  
17 // Here we're starting the loop, instructing it  
18 // to run until there are no more sources.  
loop.run();
```

If you run that example as a script it would never exit, because the loop is never instructed to quit. The example below will iterate the main context for one second, before instructing the loop to quit:

```
1 import GLib from 'gi://GLib';  
2  
3 // Here we're starting the loop, instructing it  
4 // to run until there are no more sources.  
loop.run();
```



GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

```
4 const loop = new GLib.MainLoop(null, false);
5
6 const sourceId = GLib.timeout_add_seconds(1, () => {
7     loop.quit();
8
9     return GLib.SOURCE_REMOVE;
10 });
11
12 log('Starting the main loop');
13
14 // This function will return when GLib.MainLoop.run() is called
15 loop.run();
16
17 log('The main loop stopped');
```

Event Sources

There are many types of sources in GLib and you may even create your own, but the two most common you will create explicitly are timeout sources and idle sources:

```
1 import GLib from 'gi://GLib';
2
3
4 const loop = new GLib.MainLoop(null, false);
5
6 // Timeout sources execute a callback when a specified time has passed
7 const timeoutId = GLib.timeout_add_seconds(1, () => {
8     console.log('This callback was invoked');
9
10    return GLib.SOURCE_REMOVE;
11 });
12
13
14 // Idle sources execute a callback when a file or resource becomes ready.
15
```

GJS[Intro](#)**Asynchronous Programming**[The Main Loop](#)[Event Sources](#)[Event Priority](#)[Removing Sources](#)[Promises](#)[Traditional Usage](#)[async/await](#)[Asynchronous Operations](#)[Traditional Usage](#)[async/await](#)[Promisify Helper](#) [Cancelling Operations](#)[Style Guide](#)[Tips On Memory Management](#)**GLib**[GVariant](#)**GObject**[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)**Gio**[Actions and Menus](#)

```
19     return GLib.SOURCE_REMOVE;
20   });
21
22   loop.run();
```

Sources can also be created implicitly, like when calling the asynchronous methods found in Gio. These functions usually execute tasks in a background thread, then once they complete add a `GLib.Source` to the caller's main context to invoke a callback:

```
1 import GLib from 'gi://GLib';
2 import Gio from 'gi://Gio';
3
4
5 const loop = new GLib.MainLoop(null, false);
6 const file = Gio.File.new_for_path('test');
7
8 // GTask-based operations invoke a callback
9 file.delete_async(GLib.PRIORITY_DEFAULT,
10   console.log('This callback was invoked');
11
12   try {
13     file.delete_finish(result);
14   } catch (e) {
15     logError(e);
16   }
17 });
18
19 loop.run();
```

Here's one more, slightly more advanced example. In this case, we'll create a input stream for `stdin` (that's the stream you use when you type in a terminal) then use it to create a source that triggers when the user



GJS[Intro](#)**Asynchronous Programming**[The Main Loop](#)[Event Sources](#)[Event Priority](#)[Removing Sources](#)[Promises](#)[Traditional Usage](#)[async/await](#)[Asynchronous Operations](#)[Traditional Usage](#)[async/await](#)[Promisify Helper](#) [Cancelling Operations](#)[Style Guide](#)[Tips On Memory Management](#)**GLib**[GVariant](#)**GObject**[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)**Gio**[Actions and Menus](#)

```
1 import GLib from 'gi://GLib';
2 import Gio from 'gi://Gio';
3
4
5 const loop = new GLib.MainLoop(null, false);
6
7 const stdinDecoder = new TextDecoder('utf-8');
8 const stdinStream = new Gio.UnixInputStream(0);
9
10 // Here we create a GLib.Source using GIO
11 // set the priority and callback, then attach it
12 const stdinSource = stdinStream.create_source();
13 stdinSource.set_priority(GLib.PRIORITY_INPUT);
14 stdinSource.set_callback(() => {
15     try {
16         const data = stdinStream.read_bytes();
17         const text = stdinDecoder.decode(data);
18
19         print(`You typed: ${text}`);
20
21         return GLib.SOURCE_CONTINUE;
22     } catch (e) {
23         logError(e);
24
25         return GLib.SOURCE_REMOVE;
26     }
27 });
28 const sourceId = stdinSource.attach(null);
29
30 // Start processing input
31 loop.run();
```



Event Priority

Each event source will have a priority, to determine which will get "dispatched" if more than one is ready at the same time. Priorities are simply positive or negative

GJS

[Intro](#)

Asynchronous Programming

[The Main Loop](#)[Event Sources](#)[Event Priority](#)[Removing Sources](#)[Promises](#)[Traditional Usage](#)[async/await](#)[Asynchronous Operations](#)[Traditional Usage](#)[async/await](#)[Promisify Helper](#)[Cancelled Operations](#)[Style Guide](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)

Consider the table of common priorities below. In particular, notice how `Gtk.PRIORITY_RESIZE` has a higher priority than `Gdk.PRIORITY_REDRAW` so that a `Gtk.Window` isn't redrawn for every little step resizing:

| Constant | Value |
|---|-------|
| <code>GLib.PRIORITY_LOW</code> | 300 |
| <code>GLib.PRIORITY_DEFAULT_IDLE</code> | 200 |
| <code>Gdk.PRIORITY_REDRAW</code> | 120 |
| <code>Gtk.PRIORITY_RESIZE</code> | 110 |
| <code>GLib.PRIORITY_HIGH_IDLE</code> | 100 |
| <code>GLib.PRIORITY_DEFAULT</code> | 0 |
| <code>GLib.PRIORITY_HIGH</code> | -100 |

In the example below, we will add two timeout sources that resolve at the same time, but with differing priorities so that one is dispatched first:

```
1 import GLib from 'gi://GLib';
2
3
4 const loop = new GLib.MainLoop(null, false);
5
6 const idleId = GLib.timeout_add_seconds(1, () => {
7   console.log('idle source');
8
9   return GLib.SOURCE_REMOVE;
10 });
11
12 const defaultId = GLib.timeout_add_seconds(1, () =>
```



GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

```
15      return GLib.SOURCE_REMOVE,  
16  );  
17  
18  loop.run();
```

Removing Sources

There are two ways you can remove a source from the loop, if it was added with `GLib.timeout_add()` or `GLib.idle_add()`.

Both of these functions return an opaque value, just like a signal connection. This value can be passed to `GLib.Source.remove()`, after which the source will be removed from the main context and the callback will not be invoked.

The other way depends on the return value of the source callback. If the callback returns

`GLib.SOURCE_CONTINUE` the callback will be invoked again when the source's condition is met. If it returns `GLib.SOURCE_REMOVE`, the source will be removed and the callback will never be invoked again.

```
1  import GLib from 'gi://GLib';  
2  
3  
4  const loop = new GLib.MainLoop(null, false);  
5  
6  const idleId = GLib.idle_add(GLib.PRIORITY_DEFAULT,  
7      () => {  
8          console.log('This callback will only run once');  
9          return GLib.SOURCE_REMOVE;  
10     });  
11  
12  const timeoutId = GLib.timeout_add_seconds(1, () => {  
13      console.log('This callback will run every second');  
14  });
```



GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

```
15      return GLib.SOURCE_CONTINUE,
16  });
17
18  const removeId = GLib.timeout_add_seconds(1, () => {
19      console.log('This callback will be triggered once');
20      GLib.Source.remove(timeoutId);
21
22  return GLib.SOURCE_REMOVE;
23 });
24 });
25
26 loop.run();
```

Other event sources, like those created by asynchronous methods in GIO, can not be removed directly. Most operations may be cancelled however, by passing a [Gio.Cancellable](#).

Promises

In GJS, a [Promise](#) is basically an event source that triggers when the `resolve()` or `reject()` functions are invoked. If you are new to `Promise` and `async` in JavaScript, you should review the following articles on MDN:

- [Using promises](#)
- [async / await](#)

This section will only briefly cover `Promise` usage in JavaScript and GJS.



Traditional Usage

```
1 import GLib from 'gi://GLib';
```

js

GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

Variant

GObject

The Basics

Interfaces

Subclasses

Type

Value

Advanced

Gio

Actions and Menus

```
4 const loop = new GLib.MainLoop(null, false);
5
6 // Returns a Promise that randomly fails.
7 function unreliablePromise() {
8     return new Promise((resolve, reject) => {
9         GLib.timeout_add_seconds(GLib.PRIORITY_DEFAULT, () => {
10             if (Math.random() >= 0.5)
11                 resolve('success');
12             else
13                 reject(Error('failure'));
14
15             return GLib.SOURCE_REMOVE;
16         });
17     });
18 }
19
20
21 // When using a Promise in the traditional way,
22 // `then()` to get the result and `catch()` to handle errors.
23 unreliablePromise().then(result => {
24     // Logs "success"
25     console.log(result);
26 }).catch(e => {
27     // Logs "Error: failure"
28     logError(e);
29 });
30
31 // A convenient short-hand in GJS is just to use the promise
32 unreliablePromise().catch(logError);
33
34
35 loop.run();
```



async / await

Although Promise objects offer a simple API for asynchronous operations, they still have the burden of callbacks.

GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

with the benefits of asynchronous execution.

```
1 import GLib from 'gi://GLib';
2
3 const loop = new GLib.MainLoop(null, false);
4
5 // Returns a Promise that randomly fails
6 function unreliablePromise() {
7     return new Promise((resolve, reject) => {
8         GLib.timeout_add_seconds(GLib.PRIORITY_DEFAULT, () => {
9             if (Math.random() >= 0.5)
10                 resolve('success');
11             else
12                 reject(Error('failure'));
13         });
14     });
15
16     return GLib.SOURCE_REMOVE;
17 });
18 }
19
20
21 // An example async function, demonstrating
22 // sequentially while catching errors in
23 // an async function exampleAsyncFunction() {
24     try {
25         let count = 0;
26
27         while (true) {
28             await unreliablePromise();
29             console.log(`Promises resolved: ${count}`);
30         }
31     } catch (e) {
32         logError(e);
33         loop.quit();
34     }
35 }
36
37 // Run the async function
```



GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

Variant

GObject

The Basics

Interfaces

Subclasses

GType

Value

Advanced

Gio

Actions and Menus

41 loop.run();

Asynchronous Operations

While you can not run JavaScript on multiple threads, almost all GIO operations have asynchronous variants. These work by collecting the necessary input in the function arguments, sending the work to be done on another thread, then they invoke a callback in the main thread when finished.

Traditional Usage

Although written in JavaScript, the example below is how a C programmer will typically use these functions.

While this won't block the main thread for the duration of the operation, it is awkward compared to the modern programming styles usually used in high-level languages like JavaScript.

```
1 import GLib from 'gi://GLib';
2 import Gio from 'gi://Gio';
3
4 /**
5  * This callback will be invoked once the file has been loaded.
6  *
7  * @param {Gio.File} file - the file object
8  * @param {Gio.AsyncResult} result - the result of the operation
9  */
10
11 function loadContentsCb(file, result) {
12     try {
13         const [length, contents] = file.readSync(result)
```

GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

```
16          } catch (e) {
17      logError(e, `Reading ${file.get_
18      }
19  }
20
21  const file = Gio.File.new_for_path('tes-
22
23 // This method passes the file object to
24 // that thread, then invokes loadContent-
25 file.load_contents_async(GLib.PRIORITY_I
```

async / await

One of the most convenient uses of `Promise` for GJS programmers, is to wrap these asynchronous functions and use the `async / await` pattern to regain synchronous flow:

```
1 import GLib from 'gi://GLib';
2 import Gio from 'gi://Gio';
3
4
5 const file = Gio.File.new_for_path('tes-
6
7 // Here is the synchronous, blocking fo-
8 try {
9     const [, contents] = file.load_conten-
10
11     console.log(`Read ${contents.length}`);
12 } catch (e) {
13     logError(e, `Reading ${file.get_base_
14 }
15
16 // Here is an asynchronous, non-blocking
17 try {
18     const [, contents] = await new Promis-
19         file.load_contents_async(GLib.PI
```

js



GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

```
22
23         } catch (e) {
24             // If an error occurred
25             reject(e);
26         }
27     });
28 }
29
30     console.log(`Read ${contents.length}`);
31 } catch (e) {
32     logError(e, `Reading ${file.get_basename()}`);
33 }
```

The important thing to notice is that by using the `async / await` pattern, you can maintain a simple, synchronous-like programming style while taking advantage of asynchronous execution.

With a wrapper function prepared, you can even run many of these operations in parallel; each in its own thread:

```
1 import GLib from 'gi://GLib';
2 import Gio from 'gi://Gio';
3
4 /**
5  * A simple Promise wrapper that elides
6  *
7  * @param {Gio.File} file - a file object
8  * @returns {Promise<Uint8Array>} - the file's contents
9  */
10
11 function loadContents(file) {
12     return new Promise((resolve, reject) {
13         file.load_contents_async(GLib.PRIORITY_DEFAULT,
14             try {
15                 resolve(file.load_contents());
16             } catch (e) {
17                 reject(e);
18             }
19         );
20     });
21 }
```

GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

Variant

GObject

The Basics

Interfaces

Subclasses

Type

Value

Advanced

Gio

Actions and Menus

```
19
20      });
21  }
22
23  try {
24      // A list of files to read
25      const files = [
26          Gio.File.new_for_path('test-file'),
27          Gio.File.new_for_path('test-file'),
28          Gio.File.new_for_path('test-file')
29      ];
30
31      // Creating a Promise for each operation
32      const operations = files.map(file =>
33
34          // Run them all in parallel
35          const results = await Promise.all(operations);
36
37          results.forEach((result, i) => {
38              console.log(`Read ${result.length} bytes`);
39          });
40      } catch (e) {
41          logError(e);
42  }
```

Promisify Helper

In GJS 1.54 a convenient helper was added as a technology preview, based on the work of Outreachy intern [Avi Zajac](#). Ultimately, GJS will have seamless support for async functions, but until then you can use the "promisify" helper to automatically create Promise wrappers.

The `Gio._promisify()` utility replaces the original function on the class prototype, so that it can be called on any instance of the class, including subclasses.



GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

```
1 Gio._promisify(Gio.InputStream.prototype  
2   'read_bytes_finish');
```

The function may then be used like any other `Promise` without the need for a custom wrapper, simply by leaving out the callback argument:

```
1 const inputStream = new Gio.UnixInputSt  
2  
3 try {  
4   const bytes = await inputStream.read(  
5     GLib.PRIORITY_DEFAULT, null);  
6 } catch (e) {  
7   logError(e, 'Failed to read bytes')  
8 }
```

The original function will still be available, and can be used simply by passing the callback:

```
1 inputStream.read_bytes_async(  
2   4096,  
3   GLib.PRIORITY_DEFAULT,  
4   null,  
5   (stream_, result) => {  
6     try {  
7       const bytes = inputStream.re  
8     } catch (e) {  
9       logError(e, 'Failed to read  
10      }  
11    }  
12  );
```

Cancelling Operations



GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced



Gio

Actions and Menus

GJS

Intro

Asynchronous Programming

The Main Loop

Event Sources

Event Priority

Removing Sources

Promises

Traditional Usage

async/await

Asynchronous Operations

Traditional Usage

async/await

Promisify Helper

Cancelling Operations

Style Guide

Tips On Memory Management

GLib

GVariant

GObject

The Basics

Interfaces

Subclasses

GType

GValue

Advanced

Gio

Actions and Menus

```
import { Gio } from 'gi://gio';  
import { cancellable, cancellable, cancellable } from 'gi://gio';  
  
// This callback will be invoked once the file has been read.  
// It will be passed the length of the file and its contents.  
// If it was cancelled, result will be null.  
// If it was successful, result will be a Gio.AsyncResult object.  
// If it failed, result will be null and e will contain the error.  
function loadContentsCb(file, result) {  
    try {  
        const [length, contents] = file.readSync();  
        console.log(`Read ${length} bytes`);  
    } catch (e) {  
        // If the operation was cancelled,  
        // which case we may just want to ignore.  
        if (!e.matches(Gio.IOErrorEnum.CANCELLED))  
            logError(e, `Reading ${file}`);  
    }  
}  
  
const file = Gio.File.new_for_path('test.txt');  
  
// This is the cancellable we will pass  
// to the Gio.File.load_contents() method.  
const cancellable = new Gio.Cancellable();  
  
// This method passes the file object to the Gio.File.load_contents()  
// method, then invokes loadContentsCb() when it's done.  
file.load_contents_async(GLib.PRIORITY_IDLE, cancellable, loadContentsCb);  
  
// Cancel the operation by triggering the cancellable.  
cancellable.cancel();
```

You may pass the same `Gio.Cancellable` object to as many operations as you want, and cancel them all with a single call to `Gio.Cancellable.cancel()`. This can



GJS

[Intro](#)

Asynchronous Programming

[The Main Loop](#)[Event Sources](#)[Event Priority](#)[Removing Sources](#)[Promises](#)[Traditional Usage](#)[async/await](#)[Asynchronous Operations](#)[Traditional Usage](#)[async/await](#)[Promisify Helper](#) [Cancelling Operations](#)[Style Guide](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)

disabled.

Once a `Gio.Cancellable` has been cancelled, you should drop the reference to it and create a new instance for future operations.

Contributors: Last Updated: 8/23/2023, 12:08:54 PM
Andy Holmes, Andy Holmes, Marcin Jahn

[← Intro](#)[Style Guide →](#)

GJS

[Intro](#)[Asynchronous Programming](#)

Style Guide

[ESLint](#)[Continuous Integration](#)[Prettier](#)[EditorConfig](#)[Code Conventions](#)[Files and Imports](#)[GObject](#)[JavaScript](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)[Subprocesses](#)

Style Guide

This guide documents how to use the official GJS ESLint configuration, as well as other preferred styles that can't be expressed by a linter configuration.

It also includes a basic introduction to setting up a project to use `.eslintrc.yml` and `.editorconfig` files, to help reduce manual work for developers.

ESLint

TIP

GNOME Shell includes one additional global variable called [global](#).

ESLint is a well known linter and static analysis tool for JavaScript, used by both GJS and GNOME Shell. It's used by many projects to maintain code quality, enforce coding standards, catch potential errors, and improve code consistency.

The [recommended configuration](#) includes rules for static analysis, deprecated syntax and a list of all the global variables for the environment. Put the `.eslintrc.yml` in the root of your project directory, and your IDE will provide real-time diagnostics and warnings.



GJS

- [Intro](#)
- [Asynchronous Programming](#)

Style Guide

- [ESLint](#)
 - [Continuous Integration](#)
 - [Prettier](#)
 - [EditorConfig](#)
 - [Code Conventions](#)
 - [Files and Imports](#)
 - [GObject](#)
 - [JavaScript](#)
- [Tips On Memory Management](#)

GLib

- [GVariant](#)

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)
- [GType](#)
- [GValue](#)
- [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)
- [Subprocesses](#)

ESLint is transitioning to a new flat configuration that uses ES Modules. To use this configuration, be sure your project has a `package.json` file with

```
"sourceType": "module" .
```

- ▶ [eslint.config.js](#)

Continuous Integration

In most projects, it is recommended practice to run tests on every pull request before merging into the main branch. Below are two example CI configurations for running ESLint with GitLab and GitHub.

- ▶ [GitLab \(`.gitlab-ci.yml` \)](#)

- ▶ [GitHub \(`.github/workflows/eslint.yml` \)](#)

Prettier



[Prettier](#) is another popular tool for JavaScript projects, renowned for its lack of options. It focuses specifically on formatting code, and won't catch logic errors or anti-patterns like ESLint.

Below is a sample configuration (with almost all

GJS

[Intro](#)
[Asynchronous Programming](#)

Style Guide

[ESLint](#)
[Continuous Integration](#)

[Prettier](#)

[EditorConfig](#)

[Code Conventions](#)

[Files and Imports](#)

[GObject](#)

[JavaScript](#)

[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)

[Interfaces](#)

[Subclasses](#)

[GType](#)

[GValue](#)

[Advanced](#)

Gio

[Actions and Menus](#)

[D-Bus](#)

[File Operations](#)

[List Models](#)

[Subprocesses](#)

```
1  tabWidth: 4
2  useTabs: false
3  semi: true
4  singleQuote: true
5  quoteProps: 'as-needed'
6  trailingComma: 'es5'
7  bracketSpacing: false
8  arrowParens: 'avoid'
```

yml

EditorConfig

[EditorConfig](#) is a more general formatting tool, targeted directly at IDEs like GNOME Builder and VSCode. It's used to tell an editor to trim trailing whitespace, what indentation to use, and other similar preferences.

Below is the `.editorconfig` file used in the GJS project:

```
1  root = true
2
3  [*]
4  indent_style = space
5  indent_size = 4
6  charset = utf-8
7  trim_trailing_whitespace = true
8  end_of_line = lf
9  insert_final_newline = true
10
11 [*.*]
12 quote_type = single
```



GJS

- [Intro](#)
- [Asynchronous Programming](#)
- Style Guide**
 - [ESLint](#)
 - [Continuous Integration](#)
 - [Prettier](#)
 - [EditorConfig](#)
 - [Code Conventions](#)
 - [Files and Imports](#)
 - [GObject](#)
 - [JavaScript](#)
- [Tips On Memory Management](#)

GLib

- [GVariant](#)

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)
- [GType](#)
- [GValue](#)
- [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)
- [Subprocesses](#)

Code Conventions

The following guidelines are general recommendations and coding conventions followed by many GJS projects. As general rule, you should take advantage of modern language features, both in JavaScript and GJS.

Files and Imports

TIP

GJS has supported ESModules since GNOME 40, and GNOME Shell extensions are required to use them since GNOME 45.

JavaScript file names should be `lowerCamelCase` with a `.js` extension, while directories should be short and lowercase :

```
js/misc/extensionSystem.js  
js/ui/panel.js
```

sh

Use `PascalCase` when importing modules and classes

```
import * as Util from 'resource:///gjs/guide/
```



Keep library, module and local imports separated by a single line.

```
import Gio from 'gi://Gio';
```

js

GJS

[Intro](#)[Asynchronous Programming](#)

Style Guide

[ESLint](#)[Continuous Integration](#)[Prettier](#)[EditorConfig](#)[Code Conventions](#)[Files and Imports](#)[GObject](#)[JavaScript](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)[Subprocesses](#)

```
import * as Util from './lib/util.js';
```

GObject

TIP

See [GObject Basics](#) for more details about using GObject in JavaScript.

Properties

When possible, set all properties when constructing an object, which is cleaner and avoids extra property notifications.

```
const label = new Gtk.Label({  
    label: 'Example',  
});
```

js

Using `camelCase` property accessors is preferred by many GNOME projects. GJS can automatically convert GObject property names, except when used as a string.

```
label.useMarkup = true;
```

js

```
label.bind_property('use-markup', label, 'use  
GObject.BindFlags.SYNC_CREATE | GObject.B
```



Asynchronous Operations

Use `Gio._promisify()` to enable `async / await`

js

GJS

- [Intro](#)
- [Asynchronous Programming](#)
- Style Guide**
 - [ESLint](#)
 - [Continuous Integration](#)
 - [Prettier](#)
 - [EditorConfig](#)
 - [Code Conventions](#)
 - [Files and Imports](#)
 - [GObject](#)
 - [JavaScript](#)
- [Tips On Memory Management](#)

GLib

- [GVariant](#)
- GObject**
 - [The Basics](#)
 - [Interfaces](#)
 - [Subclasses](#)
 - [GType](#)
 - [GValue](#)
 - [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)
- [Subprocesses](#)

```
import GLib from 'gi://GLib';
import Gio from 'gi://Gio';

Gio._promisify(Gio.File.prototype, 'delete_asynchronous')
const file = Gio.File.new_for_path('file.txt')
await file.delete_async(GLib.PRIORITY_DEFAULT)
```

JavaScript

Variables and Exports

Use `const` when the value will be bound to a static value, and `let` when you need a mutable variable:

```
const elementCount = 10;
const elements = [];

for (let i = 0; i < elementCount; i++)
  elements.push(i);

for (const element of elements)
  console.log(`Element ${element + 1}`);
```

The `var` statement should be avoided, since it has unexpected behavior like [hoisting](#). Although it was used in older code to make members of a script public, `export` should now be used in all new code:



```
export const PUBLIC_CONSTANT = 100;

export const PublicObject = GObject.registerClass(
  class PublicObject extends GObject.Object {
    frobnicate() {
    }
  }
);
```

GJS

- [Intro](#)
- [Asynchronous Programming](#)
- Style Guide**
 - [ESLint](#)
 - [Continuous Integration](#)
 - [Prettier](#)
 - [EditorConfig](#)
 - [Code Conventions](#)
 - [Files and Imports](#)
 - [GObject](#)
 - [JavaScript](#)
- [Tips On Memory Management](#)

GLib

- [GVariant](#)

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)
- [GType](#)
- [GValue](#)
- [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)
- [Subprocesses](#)

Classes and Functions

Define classes with `class` and override the standard `constructor()` when subclassing GObject classes:

```
class MyObject {  
    frobnicate() {  
    }  
}  
  
const MySubclass = GObject.registerClass(  
class MySubclass extends GObject.Object {  
    constructor(params = {}) {  
        /* Chain-up with an object of constru  
        super(params);  
    }  
  
    frobnicate() {  
    }  
});
```

Use `arrow functions` for inline callbacks and `Function.prototype.bind()` for larger functions.

```
class MyClock {  
    constructor() {  
        this._settings = new Gio.Settings({  
            schema_id: 'org.gnome.desktop.in  
        });  
  
        this._settings.connect('changed::cloc  
        this.showSeconds = this._settings  
    });  
  
    this._settings.connect('changed::cloc
```

GJS

[Intro](#)[Asynchronous Programming](#)

Style Guide

[ESLint](#)[Continuous Integration](#)[Prettier](#)[EditorConfig](#)[Code Conventions](#)[Files and Imports](#)[GObject](#)[JavaScript](#)[Tips On Memory Management](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)[Subprocesses](#)

```
_onShowWeekdaysChanged() {
    this.showWeekdays = this._settings.get('show-weekdays');
}
```

Contributors: Last Updated: 11/16/2023, 9:05:19 AM
Andy Holmes, Andy Holmes, Ryann Ferreira, Evan Welsh, Evan Welsh, GHOST, Pedro Sader Azevedo

[← Asynchronous Programming](#)[Tips On Memory Management →](#)

GJS

- [Intro](#)
- [Asynchronous Programming](#)
- [Style Guide](#)
- [**Tips On Memory Management**](#)
 - [Basics](#)
 - [Relevant Reading](#)
 - [Examples](#)
 - [Use After Free](#)
 - [Leaking References](#)
 - [Scope](#)
 - [Main Loop Sources](#)
 - [Signal Callbacks](#)
 - [Cairo](#)

GLib

- [GVariant](#)

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)
- [GType](#)
- [GValue](#)
- [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)

Tips On Memory Management

GJS is JavaScript bindings for GNOME, which means that behind the scenes there are two types of memory management happening: reference tracing (JavaScript) and reference counting (GObject).

Most developers will never have to worry about GObject referencing or memory leaks, especially if writing clean, uncomplicated code. This page describes some common ways developers fail to take scope into account or cleanup main loop sources and signal connections.

Basics

The concept of reference counting is very simple. When a GObject is first created, it has a reference count of 1 . When the reference count drops to 0 , all the object's resources and memory are automatically freed.

The concept of reference tracing is also quite simple, but can get confusing because it relies on external factors. When a value or object is no longer assigned to any variable, it will be garbage collected. In other words, if the JavaScript engine can not "trace" a value back to a variable it will free that value.

Put simply, as long as GJS can trace a GObject to a variable, it will ensure the reference count does not

GJS

- [Intro](#)
- [Asynchronous Programming](#)
- [Style Guide](#)
- ### Tips On Memory Management

 - [Basics](#)
 - [Relevant Reading](#)
 - [Examples](#)
 - [Use After Free](#)
 - [Leaking References](#)
 - [Scope](#)
 - [Main Loop Sources](#)
 - [Signal Callbacks](#)
 - [Cairo](#)

GLib

- [GVariant](#)

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)
- [GType](#)
- [GValue](#)
- [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)

Relevant Reading

If you are very new to programming, you should familiarize yourself with the concept of **scope** and the three types of variables in JavaScript: `const`, `let` and `var`. A basic understanding of these will help you a lot while trying to ensure the garbage collector can do its job.

You should also consider enabling **strict mode** in all your scripts, as this will prevent some mistakes that lead to uncollectable objects.

- [const](#)
- [let](#)
- [var](#)
- [Variable Scope](#)
- [Strict mode](#)

Examples

Below we create a `GtkLabel` and assign it to the variable `myLabel`. By assigning it to a variable we are "tracing" a reference to it, preventing the JS Object from being garbage collected and the GObject from being freed:

```
const myLabel = new Gtk.Label({  
    label: 'Some Text',  
});
```

js



Because `let` has block scope, `myLabel` would stop being traced when the scope is left. If the GObject is not assigned to a variable in a parent scope, it will be collected and freed.

GJS

- [Intro](#)
- [Asynchronous Programming](#)
- [Style Guide](#)

Tips On Memory Management

- [Basics](#)
- [Relevant Reading](#)
- [Examples](#)
- [Use After Free](#)
- [Leaking References](#)
- [Scope](#)
- [Main Loop Sources](#)
- [Signal Callbacks](#)

Cairo

GLib

GVariant

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)
- [GType](#)
- [GValue](#)
- [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)

```
// another value or the script exits
let myLabel = null;

if (myLabel === null) {
    // This variable is only valid inside this block
    const myLabelScoped = new Gtk.Label({
        label: 'Some Text',
    });

    // After assigning `myLabelScoped` to `myLabel`, it will
    // be traced from two variables, preventing it from being freed
    myLabel = myLabelScoped;
}

// The GObject is no longer being traced from
// traced from `myLabel` so it will not be collected
console.log(myLabel.label);

// After we set `myLabel` to `null`, it will
// no longer be traced from any variable, thus it will be collected and the
// memory will be freed
myLabel = null;
```

Other GObjects can hold a reference to GObjects, such as a container object. This means that even if it can not be traced from a JavaScript variable, it will have a positive reference count and not be freed.

```
let myLabel = new Gtk.Label({
    label: 'Some Text',
});
```



```
// Once we add `myLabel` to `myFrame`, the GObject
// will have a reference to it, increasing its reference
// count and preventing it from being freed, as it
// is now traced from a JavaScript variable
const myFrame = new Gtk.Frame();
myFrame.set_child(myLabel);
```

GJS

[Intro](#)[Asynchronous Programming](#)[Style Guide](#)

Tips On Memory Management

[Basics](#)[Relevant Reading](#)[Examples](#)[Use After Free](#)[Leaking References](#)[Scope](#)[Main Loop Sources](#)[Signal Callbacks](#)[Cairo](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)

```
// It is NOT necessary to do this in most cases  
// tracing the GObject when it falls out of scope  
myLabel = null;
```

However, if the only thing preventing a GObject from being collected is another GObject holding a reference, once it drops that reference it will be freed.

```
const myFrame = new Gtk.Frame();  
  
if (myFrame) {  
    const myLabel = new Gtk.Label({  
        label: 'Some Text',  
    });  
  
    myFrame.set_child(myLabel);  
}
```

```
// Even though it has not been explicitly destroyed,  
// GtkWidget will drop to 0 and the GObject will be freed.  
myFrame.set_child(null);
```

Use After Free

Most functions that affect memory management are not available in GJS, but there are some exceptions. Functions like `Clutter.Actor.destroy()` found in some libraries will force a GObject to be freed as though its reference count had dropped to `0`. What it won't do is stop a JavaScript variable from tracing that GObject.

Attempting to access a GObject after it has been finalized (freed) is a programmer's error that is not



GJS

[Intro](#)[Asynchronous Programming](#)[Style Guide](#)

Tips On Memory Management

[Basics](#)[Relevant Reading](#)[Examples](#)[Use After Free](#)[Leaking References](#)[Scope](#)[Main Loop Sources](#)[Signal Callbacks](#)[Cairo](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)

the log. Below is a simple example of how you can continue to trace a reference to a GObject that has already been freed:

```
let myLabel = new St.Label({
  text: 'Some Text',
});

// Here we are FORCING the GObject to be freed
// count had dropped to 0.
myLabel.destroy();

// Even though this GObject is being traced if
// its methods or access its properties will
// all its resources have been freed.
console.log(myLabel.text);

// In this case we are in the top-level scope
// `null` the variable to allow garbage collection
myLabel = null;
```

Leaking References

Although memory is managed for you by GJS, remember that this is done by tracing to variables. If you lose track of a variable, the ID for a signal or source callback you will leak that reference.



Scope

The easiest way to leak references is to overwrite a variable or let it fall out of scope. If this variable points to a GObject with a positive reference count that you are responsible for freeing, you will effectively leak its

GJS

[Intro](#)[Asynchronous Programming](#)[Style Guide](#)

Tips On Memory Management

[Basics](#)[Relevant Reading](#)[Examples](#)[Use After Free](#)[Leaking References](#)[Scope](#)[Main Loop Sources](#)[Signal Callbacks](#)[Cairo](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)

The following example demonstrates how reference leaks often occur in [GNOME Shell Extensions](#). When the `enable()` function returns the ID to the `indicator` object is collected, and we have lost our ability to destroy the object when `disable()` is called.

```
1 import GLib from 'gi://GLib';
2
3 import {Extension} from 'resource:///org/gtk/gjs/api';
4 import * as Main from 'resource:///org/gnome/desktop/main/ui/status-area.js';
5 import * as PanelMenu from 'resource:///org/gnome/desktop/status/icon-menu.js';
6
7
8 export default class ExampleExtension extends Extension {
9   enable() {
10     const indicator = new PanelMenu.Indicator();
11     const randomId = GLib.uuid_random().toString();
12
13     Main.panel.addToStatusArea(randomId, indicator);
14   }
15
16   disable() {
17     // When `enable()` returned, both
18     // `this` and `indicator` were in scope and got collected. However,
19     // `indicator` still had a reference to `this` via its parent
20     // `PanelMenu.Indicator` object.
21     // Every time the extension is disabled, it creates
22     // a new, unremovable indicator.
23   }
24 }
```



Here's another example. Although the principle of the code below is sound, we are leaking a reference to a `GObject` we are responsible for (and thus memory). The leak below is fairly easy to spot; what's important is how and why that reference was leaked. Mistakes like these are often easier to make and harder to track.

GJS

[Intro](#)[Asynchronous Programming](#)[Style Guide](#)

Tips On Memory Management

[Basics](#)[Relevant Reading](#)[Examples](#)[Use After Free](#)[Leaking References](#)[Scope](#)[Main Loop Sources](#)[Signal Callbacks](#)[Cairo](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)

```
1 import GLib from 'gi://GLib';
2
3 import {Extension} from 'resource:///org/gtk/gjs/test';
4 import * as Main from 'resource:///org/gtk/gjs/test/main';
5 import * as PanelMenu from 'resource:///org/gtk/gjs/test/panelmenu';
6
7
8 export default class ExampleExtension extends Extension {
9   enable() {
10     this._indicators = {};
11
12     const indicator1 = new PanelMenu.Indicator();
13     const indicator2 = new PanelMenu.Indicator();
14
15     Main.panel.addStatusArea(GLib.PanelStatus.INDICATOR);
16     Main.panel.addStatusArea(GLib.PanelStatus.INDICATOR);
17
18     this._indicators['MyIndicator1'] = indicator1;
19     this._indicators['MyIndicator2'] = indicator2;
20   }
21
22   disable() {
23     for (const [name, indicator] of Object.entries(this._indicators)) {
24       indicator.destroy();
25     }
26     this._indicators = {};
27   }
28 }
```

► Having trouble to find the leak?



Main Loop Sources

A very common way of leaking GSource's is recursive (repeating) sources added to the GLib event loop. These

GJS

[Intro](#)[Asynchronous Programming](#)[Style Guide](#)

Tips On Memory Management

[Basics](#)[Relevant Reading](#)[Examples](#)[Use After Free](#)[Leaking References](#)[Scope](#)[Main Loop Sources](#)[Signal Callbacks](#)[Cairo](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)

In the example [GNOME Shell Extension](#) below, the ID required to remove the `GSource` from the main loop has been lost and the callback will continue to be invoked even after the object has been destroyed.

So in this case, the leak is caused by the main loop holding a reference to the `GSource`, while the programmer has lost their ability to remove it. When the source callback is invoked, it will try to access the object after it has been destroyed, causing a critical error.

```
1 import GLib from 'gi://GLib';
2 import GObject from 'gi://GObject';
3
4 import {Extension} from 'resource:///org/gnome/gjs/extension.js';
5 import * as PanelMenu from 'resource:///org/gnome/gjs/ui/panel-menu.js';
6
7
8 const MyIndicator = GObject.registerClass({
9   class MyIndicator extends PanelMenu.Button {
10     startUpdating() {
11       // When `startUpdating()` returns, we have lost the
12       // `sourceId`, so we will be unable to remove it.
13       const sourceId = GLib.timeout_add(100, () =>
14         this.update.bind(this));
15     }
16
17     update() {
18       // If the object has been destroyed, `this` is
19       // undefined, so we can't call `Function.bind()`.
20       // The use of Function.bind() is safe here because
21       // `this` is a JavaScript object, regardless of its
22       // C pointer.
23       this.visible = !this.visible;
24
25       // Returning `true` or `GLib.SOURCE_CONTINUE`
26       // persist, so the callback will be invoked again.
```



GJS

- [Intro](#)
- [Asynchronous Programming](#)
- [Style Guide](#)

Tips On Memory Management

- [Basics](#)
- [Relevant Reading](#)
- [Examples](#)
- [Use After Free](#)
- [Leaking References](#)
- [Scope](#)
- [Main Loop Sources](#)
- [Signal Callbacks](#)
- [Cairo](#)

GLib

- [GVariant](#)

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)

GType**GValue****Advanced****Gio**

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)

```
29     _onDestroy() {
30         // We don't have a reference to
31         // source from the loop. We shou
32         // object property in `__init__()
33
34         super._onDestroy();
35     }
36 );
37
38
39 export default class ExampleExtension e;
40     enable() {
41         this._indicator = new MyIndicator();
42
43         // Each time the extension is en
44         // main loop by this function
45         this._indicator.startUpdating();
46     }
47
48     disable() {
49         this._indicator.destroy();
50         this._indicator = null;
51
52         // Even though we have destroyed
53         // the `indicator` variable, the
54         // invoked every 5 seconds.
55     }
56 }
```

Signal Callbacks



Like main loop sources, whenever a signal is connected it returns a handler ID. Failing to remove sources or disconnect signals can result in "use-after-free" scenarios, but it's also possible to trace variables and leak references in a callback.

GJS

[Intro](#)[Asynchronous Programming](#)[Style Guide](#)

Tips On Memory Management

[Basics](#)[Relevant Reading](#)[Examples](#)[Use After Free](#)[Leaking References](#)[Scope](#)[Main Loop Sources](#)[Signal Callbacks](#)[Cairo](#)

GLib

[GVariant](#)

GObject

[The Basics](#)[Interfaces](#)[Subclasses](#)[GType](#)[GValue](#)[Advanced](#)

Gio

[Actions and Menus](#)[D-Bus](#)[File Operations](#)[List Models](#)

dictionary, which is why the callback continues to work after `constructor()` returns. In fact, even after we set `myObject` to `null` in `disable()`, the dictionary is still be traced from within the signal callback.

```
1 import Gio from 'gi://Gio';  
2  
3 import {Extension} from 'resource:///org/gnome/desktop/integration/extensions.js';  
4  
5 const mySettings = new Gio.Settings({  
6     schema_id: 'org.gnome.desktop.integration.extensions'  
7 });  
8  
9 class MyObject {  
10    constructor() {  
11        const agent = {  
12            'family': 'Bond',  
13            'given': 'James',  
14            'codename': '007',  
15            'licensed': true,  
16        };  
17  
18        // Here is where we should have  
19        const id = mySettings.connect('changed::agent',  
20            // Here we are tracing a reference to  
21            // the signal handler ID is  
22            // `agent` dictionary is leaked  
23            // because it's not being  
24            // released by the signal handler  
25            agent.licensed = mySettings.  
26        );  
27    }  
28  
29  
30    export default class ExampleExtension extends Extension {  
31        enable() {  
32            this._myObject = new MyObject()  
33        }  
34    }  
35}
```



GJS

- [Intro](#)
- [Asynchronous Programming](#)
- [Style Guide](#)

Tips On Memory Management

- [Basics](#)
- [Relevant Reading](#)
- [Examples](#)
- [Use After Free](#)
- [Leaking References](#)
- [Scope](#)
- [Main Loop Sources](#)
- [Signal Callbacks](#)
- [Cairo](#)

GLib

- [GVariant](#)

GObject

- [The Basics](#)
- [Interfaces](#)
- [Subclasses](#)
- [GType](#)
- [GValue](#)
- [Advanced](#)

Gio

- [Actions and Menus](#)
- [D-Bus](#)
- [File Operations](#)
- [List Models](#)

```
36      this._myObject = null;
37  }
38 }
```

Cairo

Cairo is an exception in GJS, in that it is necessary to manually free the memory of a `CairoContext` at the end of a drawing method. This is simple to do, but forgetting to do it can leak a lot of memory because widgets can be redrawn quite often.

```
// Create a drawing area and set a drawing function
const drawingArea = new Gtk.DrawingArea();

drawingArea.set_draw_func((area, cr, _width,
                           _height) => {
    // Perform operations on the surface context

    // Freeing the context before returning from the function
    cr.$dispose();
});
```

Contributors: Last Updated: 9/10/2023, 5:09:39 PM
Andy Holmes, Andy Holmes, Javad Rahmatzadeh, Marcin Jahn

[← Style Guide](#)



Option Declarations

An option declaration specifies the name, type and description of a NixOS configuration option. It is invalid to define an option that hasn't been declared in any module. An option declaration generally looks like this:

```
options = {
  name = mkOption {
    type = type specification;
    default = default value;
    example = example value;
    description = lib.mdDoc "Description for use in the NixOS manual.";
  };
};
```

The attribute names within the `name` attribute path must be camel cased in general but should, as an exception, match the package attribute name when referencing a Nixpkgs package. For example, the option `services.nix-serve.bindAddress` references the `nix-serve` Nixpkgs package.

The function `mkOption` accepts the following arguments.

type The type of the option (see). This argument is mandatory for nixpkgs modules. Setting this is highly recommended for the sake of documentation and type checking. In case it is not set, a fallback type with unspecified behavior is used.

default The default value used if no value is defined by any module. A default is not required; but if a default is not given, then users of the module will have to define the value of the option, otherwise an error will be thrown.

defaultText A textual representation of the default value to be rendered verbatim in the manual. Useful if the default value is a complex expression or depends on other values or packages. Use `lib.literalExpression` for a Nix expression, `lib.literalMD` for a plain English description in Nixpkgs-flavored Markdown format.

example An example value that will be shown in the NixOS manual. You can use `lib.literalExpression` and `lib.literalMD` in the same way as in `defaultText`.

description A textual description of the option, in Nixpkgs-flavored Markdown format, that will be included in the NixOS manual. During the migration process from DocBook it is necessary to mark descriptions written in CommonMark with `lib.mdDoc`. The description may still be written in DocBook (without any marker), but this is discouraged and will be deprecated in the future.

Utility functions for common option patterns

`mkEnableOption`

Creates an Option attribute set for a boolean value option i.e an option to be toggled on or off.

This function takes a single string argument, the name of the thing to be toggled.

The option's description is “Whether to enable <name>.”.

For example:

```
mkEnableOption usage

lib.mkEnableOption (lib.mdDoc "magic")
# is like
lib.mkOption {
  type = lib.types.bool;
  default = false;
  example = true;
  description = lib.mdDoc "Whether to enable magic.";
}
```

`mkPackageOption`

Usage:

```
mkPackageOption pkgs "name" { default = [ "path" "in" "pkgs" ]; example = "literal example"; }
```

Creates an Option attribute set for an option that specifies the package a module should use for some purpose.

Note: You shouldn't necessarily make package options for all of your modules. You can always overwrite a specific package throughout nixpkgs by using nixpkgs overlays.

The package is specified in the third argument under `default` as a list of strings representing its attribute path in nixpkgs (or another package set). Because of this, you need to pass nixpkgs itself (or a subset) as the first argument.

The second argument may be either a string or a list of strings. It provides the display name of the package in the description of the generated option (using only the last element if the passed value is a list) and serves as the fallback value for the `default` argument.

To include extra information in the description, pass `extraDescription` to append arbitrary text to the generated description. You can also pass an `example` value, either a literal string or an attribute path.

The `default` argument can be omitted if the provided name is an attribute of `pkgs` (if name is a string) or a valid attribute path in `pkgs` (if name is a list).

If you wish to explicitly provide no default, pass `null` as `default`.

Examples:

Simple `mkPackageOption` usage

```
lib.mkPackageOption pkgs "hello" {}  
# is like  
lib.mkOption {  
    type = lib.types.package;  
    default = pkgs.hello;  
    defaultText = lib.literalExpression "pkgs.hello";  
    description = lib.mdDoc "The hello package to use.";  
}
```

`mkPackageOption` with explicit default and example

```
lib.mkPackageOption pkgs "GHC" {  
    default = [ "ghc" ];  
    example = "pkgs.haskell.packages.ghc92.ghc.withPackages (hkg: [ hkg.primes ]);"  
}  
# is like  
lib.mkOption {  
    type = lib.types.package;  
    default = pkgs.ghc;  
    defaultText = lib.literalExpression "pkgs.ghc";  
    example = lib.literalExpression "pkgs.haskell.packages.ghc92.ghc.withPackages (hkg: [ hkg);  
    description = lib.mdDoc "The GHC package to use."  
}
```

`mkPackageOption` with additional description text

```
mkPackageOption pkgs [ "python39Packages" "pytorch" ] {  
    extraDescription = "This is an example and doesn't actually do anything.";  
}  
# is like  
lib.mkOption {  
    type = lib.types.package;  
    default = pkgs.python39Packages.pytorch;  
    defaultText = lib.literalExpression "pkgs.python39Packages.pytorch";  
    description = "The pytorch package to use. This is an example and doesn't actually do any"  
}
```

Extensible Option Types

Extensible option types is a feature that allow to extend certain types declaration through multiple module files. This feature only work with a restricted set

of types, namely `enum` and `submodules` and any composed forms of them.

Extensible option types can be used for `enum` options that affects multiple modules, or as an alternative to related `enable` options.

As an example, we will take the case of display managers. There is a central display manager module for generic display manager options and a module file per display manager backend (`sddm`, `gdm` ...).

There are two approaches we could take with this module structure:

- Configuring the display managers independently by adding an enable option to every display manager module backend. (NixOS)
- Configuring the display managers in the central module by adding an option to select which display manager backend to use.

Both approaches have problems.

Making backends independent can quickly become hard to manage. For display managers, there can only be one enabled at a time, but the type system cannot enforce this restriction as there is no relation between each backend's `enable` option. As a result, this restriction has to be done explicitly by adding assertions in each display manager backend module.

On the other hand, managing the display manager backends in the central module will require changing the central module option every time a new backend is added or removed.

By using extensible option types, it is possible to create a placeholder option in the central module (Example: Extensible type placeholder in the service module), and to extend it in each backend module (Example: Extending `services.xserver.displayManager.enable` in the `gdm` module, Example: Extending `services.xserver.displayManager.enable` in the `sddm` module).

As a result, `displayManager.enable` option values can be added without changing the main service module file and the type system automatically enforces that there can only be a single display manager enabled.

Extensible type placeholder in the service module

```
services.xserver.displayManager.enable = mkOption {
  description = "Display manager to use";
  type = with types; nullOr (enum [ ]);
};
```

Extending `services.xserver.displayManager.enable` in the `gdm` module

```
services.xserver.displayManager.enable = mkOption {
  type = with types; nullOr (enum [ "gdm" ]);
};
```

Extending services.xserver.displayManager.enable in the sddm module

```
services.xserver.displayManager.enable = mkOption {  
    type = with types; nullOr (enum [ "sddm" ]);  
};
```

The placeholder declaration is a standard `mkOption` declaration, but it is important that extensible option declarations only use the `type` argument.

Extensible option types work with any of the composed variants of `enum` such as `with types; nullOr (enum ["foo" "bar"])` or `with types; listOf (enum ["foo" "bar"])`.

Option Definitions

Option definitions are generally straight-forward bindings of values to option names, like

```
config = {
    services.httpd.enable = true;
};
```

However, sometimes you need to wrap an option definition or set of option definitions in a *property* to achieve certain effects:

Delaying Conditionals

If a set of option definitions is conditional on the value of another option, you may need to use `mkIf`. Consider, for instance:

```
config = if config.services.httpd.enable then {
    environment.systemPackages = [ ... ];
    ...
} else {};
```

This definition will cause Nix to fail with an “infinite recursion” error. Why? Because the value of `config.services.httpd.enable` depends on the value being constructed here. After all, you could also write the clearly circular and contradictory:

```
config = if config.services.httpd.enable then {
    services.httpd.enable = false;
} else {
    services.httpd.enable = true;
};
```

The solution is to write:

```
config = mkIf config.services.httpd.enable {
    environment.systemPackages = [ ... ];
    ...
};
```

The special function `mkIf` causes the evaluation of the conditional to be “pushed down” into the individual definitions, as if you had written:

```
config = {
    environment.systemPackages = if config.services.httpd.enable then [ ... ] else [];
    ...
};
```

Setting Priorities

A module can override the definitions of an option in other modules by setting an *override priority*. All option definitions that do not have the lowest priority value are discarded. By default, option definitions have priority 100 and option defaults have priority 1500. You can specify an explicit priority by using `mkOverride`, e.g.

```
services.openssh.enable = mkOverride 10 false;
```

This definition causes all other definitions with priorities above 10 to be discarded. The function `mkForce` is equal to `mkOverride` 50, and `mkDefault` is equal to `mkOverride` 1000.

Ordering Definitions

It is also possible to influence the order in which the definitions for an option are merged by setting an *order priority* with `mkOrder`. The default order priority is 1000. The functions `mkBefore` and `mkAfter` are equal to `mkOrder` 500 and `mkOrder` 1500, respectively. As an example,

```
hardware.firmware = mkBefore [ myFirmware ];
```

This definition ensures that `myFirmware` comes before other unordered definitions in the final list value of `hardware.firmware`.

Note that this is different from override priorities: setting an order does not affect whether the definition is included or not.

Merging Configurations

In conjunction with `mkIf`, it is sometimes useful for a module to return multiple sets of option definitions, to be merged together as if they were declared in separate modules. This can be done using `mkMerge`:

```
config = mkMerge
[ # Unconditional stuff.
{ environment.systemPackages = [ ... ];
}
# Conditional stuff.
(mkIf config.services.bla.enable {
    environment.systemPackages = [ ... ];
})
];
```

Options Types

Option types are a way to put constraints on the values a module option can take. Types are also responsible of how values are merged in case of multiple value definitions.

Basic types

Basic types are the simplest available types in the module system. Basic types include multiple string types that mainly differ in how definition merging is handled.

types.bool A boolean, its values can be `true` or `false`. All definitions must have the same value, after priorities. An error is thrown in case of a conflict.

types.boolByOr A boolean, its values can be `true` or `false`. The result is `true` if *any* of multiple definitions is `true`. In other words, definitions are merged with the logical *OR* operator.

types.path A filesystem path is anything that starts with a slash when coerced to a string. Even if derivations can be considered as paths, the more specific **types.package** should be preferred.

types.pathInStore A path that is contained in the Nix store. This can be a top-level store path like `pkgs.hello` or a descendant like `"${pkgs.hello}/bin/hello"`.

types.package A top-level store path. This can be an attribute set pointing to a store path, like a derivation or a flake input.

types.enum One element of the list *l*, e.g. `types.enum ["left" "right"]`. Multiple definitions cannot be merged.

types.anything A type that accepts any value and recursively merges attribute sets together. This type is recommended when the option type is unknown.

types.anything

Two definitions of this type like

```
{  
    str = lib.mkDefault "foo";  
    pkg.hello = pkgs.hello;  
    fun.fun = x: x + 1;  
}  
  
{  
    str = lib.mkIf true "bar";  
    pkg.gcc = pkgs.gcc;
```

```
    fun.fun = lib.mkForce (x: x + 2);
}
```

will get merged to

```
{
  str = "bar";
  pkg.gcc = pkgs.gcc;
  pkg.hello = pkgs.hello;
  fun.fun = x: x + 2;
}
```

types.raw A type which doesn't do any checking, merging or nested evaluation.

It accepts a single arbitrary value that is not recursed into, making it useful for values coming from outside the module system, such as package sets or arbitrary data. Options of this type are still evaluated according to priorities and conditionals, so `mkForce`, `mkIf` and co. still work on the option value itself, but not for any value nested within it. This type should only be used when checking, merging and nested evaluation are not desirable.

types.optionType The type of an option's type. Its merging operation ensures that nested options have the correct file location annotated, and that if possible, multiple option definitions are correctly merged together. The main use case is as the type of the `_module.freeformType` option.

types.attrs A free-form attribute set.

This type will be deprecated in the future because it doesn't recurse into attribute sets, silently drops earlier attribute definitions, and doesn't discharge `lib.mkDefault`, `lib.mkIf` and co. For allowing arbitrary attribute sets, prefer `types.attrsOf` `types.anything` instead which doesn't have these problems.

types.pkgs A type for the top level Nixpkgs package set.

Numeric types

types.int A signed integer.

types.ints.{s8, s16, s32} Signed integers with a fixed length (8, 16 or 32 bits). They go from $-2^{n/2}$ to $2^{n/2}-1$ respectively (e.g. -128 to 127 for 8 bits).

types.ints.unsigned An unsigned integer (that is ≥ 0).

types.ints.{u8, u16, u32} Unsigned integers with a fixed length (8, 16 or 32 bits). They go from 0 to 2^n-1 respectively (e.g. 0 to 255 for 8 bits).

types.ints.between lowest highest An integer between `lowest` and `highest` (both inclusive).

types.ints.positive A positive integer (that is > 0).

types.port A port number. This type is an alias to `types.ints.u16`.

types.float A floating point number.

Converting a floating point number to a string with `toString` or `toJSON` may result in precision loss.

types.number Either a signed integer or a floating point number. No implicit conversion is done between the two types, and multiple equal definitions will only be merged if they have the same type.

types.numbers.between lowest highest An integer or floating point number between *lowest* and *highest* (both inclusive).

types.numbers.nonnegative A nonnegative integer or floating point number (that is ≥ 0).

types.numbers.positive A positive integer or floating point number (that is > 0).

String types

types.str A string. Multiple definitions cannot be merged.

types.separatedString sep A string. Multiple definitions are concatenated with *sep*, e.g. `types.separatedString "|"`.

types.lines A string. Multiple definitions are concatenated with a new line "`\n`".

types.commas A string. Multiple definitions are concatenated with a comma ",".

types.envVar A string. Multiple definitions are concatenated with a colon ":".

types.strMatching A string matching a specific regular expression. Multiple definitions cannot be merged. The regular expression is processed using `builtins.match`.

Submodule types

Submodules are detailed in Submodule.

types.submodule o A set of sub options *o*. *o* can be an attribute set, a function returning an attribute set, or a path to a file containing such a value. Submodules are used in composed types to create modular options. This is equivalent to `types.submoduleWith { modules = toList o; shorthandOnlyDefinesConfig = true; }`.

`types.submoduleWith { modules, specialArgs ? {}, shorthandOnlyDefinesConfig ? false }`

Like `types.submodule`, but more flexible and with better defaults. It has parameters

- `modules` A list of modules to use by default for this submodule type. This gets combined with all option definitions to build the final list of modules that will be included.
- Only options defined with this argument are included in rendered documentation.
- `specialArgs` An attribute set of extra arguments to be passed to the module functions. The option `_module.args` should be used instead for most arguments since it allows overriding. `specialArgs` should only be used for arguments that can't go through the module fixed-point, because of infinite recursion or other problems. An example is overriding the `lib` argument, because `lib` itself is used to define `_module.args`, which makes using `_module.args` to define it impossible.
- `shorthandOnlyDefinesConfig` Whether definitions of this type should default to the `config` section of a module (see Example: Structure of NixOS Modules) if it is an attribute set. Enabling this only has a benefit when the submodule defines an option named `config` or `options`. In such a case it would allow the option to be set with `the-submodule.config = "value"` instead of requiring `the-submodule.config.config = "value"`. This is because only when modules *don't* set the `config` or `options` keys, all keys are interpreted as option definitions in the `config` section. Enabling this option implicitly puts all attributes in the `config` section.

With this option enabled, defining a non-`config` section requires using a function: `the-submodule = { ... }: { options = { ... };`.

`types.deferredModule` Whereas `submodule` represents an option tree, `deferredModule` represents a module value, such as a module file or a configuration.

It can be set multiple times.

Module authors can use its value in `imports`, in `submoduleWith`'s modules` or in `evalModules`' modules` parameter, among other places.

Note that `imports` must be evaluated before the module fixpoint. Because of this, deferred modules can only be imported into “other” fixpoints, such as submodules.

One use case for this type is the type of a “default” module that allow the user to affect all submodules in an `attrsOf submodule` at once. This

is more convenient and discoverable than expecting the module user to type-merge with the `attrsOf` submodule option.

Composed types

Composed types are types that take a type as parameter. `listOf int` and `either int str` are examples of composed types.

types.listOf t A list of *t* type, e.g. `types.listOf int`. Multiple definitions are merged with list concatenation.

types.attrsOf t An attribute set of where all the values are of *t* type. Multiple definitions result in the joined attribute set.

This type is *strict* in its values, which in turn means attributes cannot depend on other attributes. See `types.lazyAttrsOf` for a lazy version.

types.lazyAttrsOf t An attribute set of where all the values are of *t* type. Multiple definitions result in the joined attribute set. This is the lazy version of `types.attrsOf`, allowing attributes to depend on each other.

This version does not fully support conditional definitions! With an option `foo` of this type and a definition `foo.attr = lib.mkIf false 10`, evaluating `foo ? attr` will return `true` even though it should be false. Accessing the value will then throw an error. For types *t* that have an `emptyValue` defined, that value will be returned instead of throwing an error. So if the type of `foo.attr` was `lazyAttrsOf (nullOr int)`, `null` would be returned instead for the same `mkIf false` definition.

types.nullOr t `null` or type *t*. Multiple definitions are merged according to type *t*.

types.uniq t Ensures that type *t* cannot be merged. It is used to ensure option definitions are declared only once.

types.unique { message = m } t Ensures that type *t* cannot be merged. Prints the message *m*, after the line `The option <option path> is defined multiple times.` and before a list of definition locations.

types.either t1 t2 Type *t1* or type *t2*, e.g. with `types; either int str`. Multiple definitions cannot be merged.

types.oneOf [t1 t2 ...] Type *t1* or type *t2* and so forth, e.g. with `types; oneOf [int str bool]`. Multiple definitions cannot be merged.

types.coercedTo from f to Type *to* or type *from* which will be coerced to type *to* using function *f* which takes an argument of type *from* and return a value of type *to*. Can be used to preserve backwards compatibility of an option if its type was changed.

Submodule

`submodule` is a very powerful type that defines a set of sub-options that are handled like a separate module.

It takes a parameter `o`, that should be a set, or a function returning a set with an `options` key defining the sub-options. Submodule option definitions are type-checked accordingly to the `options` declarations. Of course, you can nest submodule option definitions for even higher modularity.

The option set can be defined directly (Example: Directly defined submodule) or as reference (Example: Submodule defined as a reference).

Note that even if your submodule's options all have a default value, you will still need to provide a default value (e.g. an empty attribute set) if you want to allow users to leave it undefined.

Directly defined submodule

```
options.mod = mkOption {
    description = "submodule example";
    type = with types; submodule {
        options = {
            foo = mkOption {
                type = int;
            };
            bar = mkOption {
                type = str;
            };
        };
    };
};
```

Submodule defined as a reference

```
let
    modOptions = {
        options = {
            foo = mkOption {
                type = int;
            };
            bar = mkOption {
                type = int;
            };
        };
    };
in
options.mod = mkOption {
```

```

description = "submodule example";
type = with types; submodule modOptions;
};

```

The `submodule` type is especially interesting when used with composed types like `attrsOf` or `listOf`. When composed with `listOf` (Example: Declaration of a list of submodules), `submodule` allows multiple definitions of the submodule option set (Example: Definition of a list of submodules).

Declaration of a list of submodules

```

options.mod = mkOption {
  description = "submodule example";
  type = with types; listOf (submodule {
    options = {
      foo = mkOption {
        type = int;
      };
      bar = mkOption {
        type = str;
      };
    };
  });
};

```

Definition of a list of submodules

```

config.mod = [
  { foo = 1; bar = "one"; }
  { foo = 2; bar = "two"; }
];

```

When composed with `attrsOf` (Example: Declaration of attribute sets of submodules), `submodule` allows multiple named definitions of the submodule option set (Example: Definition of attribute sets of submodules).

Declaration of attribute sets of submodules

```

options.mod = mkOption {
  description = "submodule example";
  type = with types; attrsOf (submodule {
    options = {
      foo = mkOption {
        type = int;
      };
      bar = mkOption {
        type = str;
      };
    };
  });
};

```

```
    };
  });
};
```

Definition of attribute sets of submodules

```
config.mod.one = { foo = 1; bar = "one"; };
config.mod.two = { foo = 2; bar = "two"; };
```

Extending types

Types are mainly characterized by their `check` and `merge` functions.

check The function to type check the value. Takes a value as parameter and return a boolean. It is possible to extend a type check with the `addCheck` function (Example: Adding a type check), or to fully override the check function (Example: Overriding a type check).

Adding a type check

```
byte = mkOption {
  description = "An integer between 0 and 255.";
  type = types.addCheck types.int (x: x >= 0 && x <= 255);
};
```

Overriding a type check

```
nixThings = mkOption {
  description = "words that start with 'nix'";
  type = types.str // {
    check = (x: lib.hasPrefix "nix" x)
  };
};
```

merge Function to merge the options values when multiple values are set. The function takes two parameters, `loc` the option path as a list of strings, and `defs` the list of defined values as a list. It is possible to override a type merge function for custom needs.

Custom types

Custom types can be created with the `mkOptionType` function. As type creation includes some more complex topics such as submodule handling, it is recommended to get familiar with `types.nix` code before creating a new type.

The only required parameter is `name`.

name A string representation of the type function name.

description Description of the type used in documentation. Give information of the type and any of its arguments.

check A function to type check the definition value. Takes the definition value as a parameter and returns a boolean indicating the type check result, **true** for success and **false** for failure.

merge A function to merge multiple definitions values. Takes two parameters:

loc The option path as a list of strings, e.g. `["boot" "loader "grub" "enable"]`.

defs The list of sets of defined **value** and **file** where the value was defined, e.g. `[{ file = "/foo.nix"; value = 1; } { file = "/bar.nix"; value = 2 }]`. The **merge** function should return the merged value or throw an error in case the values are impossible or not meant to be merged.

getSubOptions For composed types that can take a submodule as type parameter, this function generate sub-options documentation. It takes the current option prefix as a list and return the set of sub-options. Usually defined in a recursive manner by adding a term to the prefix, e.g. `prefix: elemType.getSubOptions (prefix ++ ["prefix"])` where `"prefix"` is the newly added prefix.

getSubModules For composed types that can take a submodule as type parameter, this function should return the type parameters submodules. If the type parameter is called **elemType**, the function should just recursively look into submodules by returning `elemType.getSubModules;`.

substSubModules For composed types that can take a submodule as type parameter, this function can be used to substitute the parameter of a submodule type. It takes a module as parameter and return the type with the submodule options substituted. It is usually defined as a type function call with a recursive call to **substSubModules**, e.g for a type **composedType** that take an **elemtype** type parameter, this function should be defined as `m: composedType (elemType.substSubModules m)`.

typeMerge A function to merge multiple type declarations. Takes the type to merge **functor** as parameter. A **null** return value means that type cannot be merged.

f The type to merge **functor**.

Note: There is a generic **defaultTypeMerge** that work with most of value and composed types.

functor An attribute set representing the type. It is used for type operations and has the following keys:

type The type function.

wrapped Holds the type parameter for composed types.

payload Holds the value parameter for value types. The types that have a **payload** are the **enum**, **separatedString** and **submodule** types.

binOp A binary operation that can merge the payloads of two same types.

Defined as a function that take two payloads as parameters and return the payloads merged.

romkatv / powerlevel10k

Code Issues Pull requests Actions Security Insights

Eye Favourite Star

A Zsh theme

MIT license

41.4k stars 2.1k forks 171 watching 5 Branches 33 Tags Activity

Public repository



master 5 Branches 33 Tags Go to file Go to file Add file Code ...

romkatv skip batteries with "Unknown" status (#2562) 3 days ago

| | | |
|----------------------------|--|--------------|
| config | add cdk to POWERLEVEL9K_AWS_SHOW_O... | 2 months ago |
| gitstatus | Merge commit 'dec881651ccbd90f7f68b2a... | 2 months ago |
| internal | skip batteries with "Unknown" status (#2562) | 3 days ago |
| .gitattributes | survive broken user-defined gitattributes | 5 years ago |
| .gitignore | zcompile all sources | 5 years ago |
| LICENSE | add a link to contributors | 5 years ago |
| Makefile | add 'minify' make target | 3 years ago |
| README.md | docs: fix a link to zsh-theme-powerlevel10k ... | 2 weeks ago |
| font.md | font: update alacritty instructions (#2539) | 3 weeks ago |
| powerlevel10k.png | update screenshot, now with rainbow style | 5 years ago |
| powerlevel10k.zsh-theme | disable re_match_pcres, otherwise we can g... | 4 years ago |
| powerlevel9k.zsh-theme | remove spurious execute permissions | 5 years ago |
| prompt_powerlevel10k_setup | replace symbolic links with regular files tha... | 5 years ago |
| prompt_powerlevel9k_setup | replace symbolic links with regular files tha... | 5 years ago |

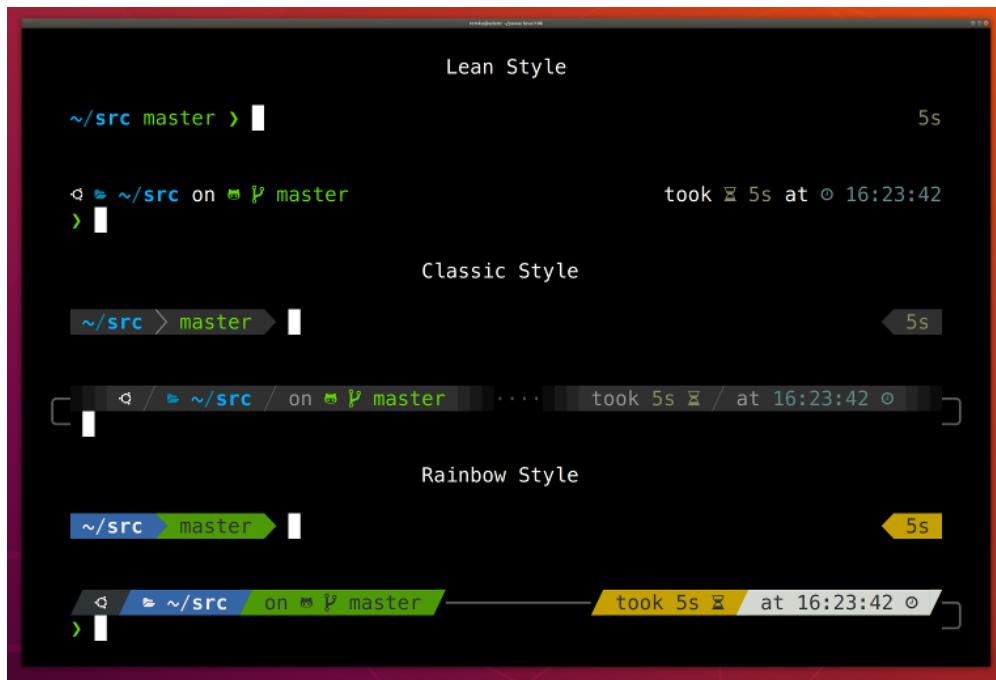
README MIT license



Powerlevel10k

chat on gitter

Powerlevel10k is a theme for Zsh. It emphasizes [speed](#), [flexibility](#) and [out-of-the-box experience](#).



- [Getting started](#)
- [Features](#)
- [Installation](#)
- [Configuration](#)
- [Fonts](#)
- [Try it in Docker](#)
- [License](#)
- [FAQ](#)
- [Troubleshooting](#)

Getting started

1. [Install the recommended font](#). *Optional but highly recommended.*
2. [Install Powerlevel10k](#) itself.
3. Restart Zsh with `exec zsh`.
4. Type `p10k configure` if the configuration wizard doesn't start automatically.

Features

- [Configuration wizard](#)
- [Uncompromising performance](#)
- [Powerlevel9k compatibility](#)
- [Pure compatibility](#)
- [Instant prompt](#)
- [Show on command](#)
- [Transient prompt](#)
- [Current directory that just works](#)
- [Extremely customizable](#)
- [Batteries included](#)
- [Extensible](#)

Configuration wizard

Type `p10k configure` to access the builtin configuration wizard right from your terminal.

► Screen recording

All styles except [Pure](#) are functionally equivalent. They display the same information and differ only in presentation.

Configuration wizard creates `~/.p10k.zsh` based on your preferences. Additional prompt customization can be done by editing this file. It has plenty of comments to help you navigate through configuration options.

Tip: Install [the recommended font](#) before running `p10k configure` to unlock all prompt styles.

FAQ:

- [What is the best prompt style in the configuration wizard?](#)
- [What do different symbols in Git status mean?](#)
- [How do I change prompt colors?](#)

Troubleshooting:

- [Some prompt styles are missing from the configuration wizard.](#)
- [Question mark in prompt.](#)
- [Icons, glyphs or powerline symbols don't render.](#)
- [Sub-pixel imperfections around powerline symbols.](#)
- [Directory is difficult to see in prompt when using Rainbow style.](#)



uncompromising performance

When you hit *ENTER*, the next prompt appears instantly. With Powerlevel10k there is no prompt lag. If you install Cygwin on Raspberry Pi, `cd` into a Linux Git repository and activate enough prompt segments to fill four prompt lines on both sides of the screen... wait, that's just crazy and no one ever does that. Probably impossible, too. The point is, Powerlevel10k prompt is always fast, no matter what you do!

- ▶ Screen recording

Note how the effect of every command is instantly reflected by the very next prompt.

| Command | Prompt Indicator | Meaning |
|---|------------------|--|
| <code>timew start hack linux</code> | ⌚ hack linux | time tracking enabled in timewarrior |
| <code>touch x y</code> | ?2 | 2 untracked files in the Git repo |
| <code>rm COPYING</code> | !1 | 1 unstaged change in the Git repo |
| <code>echo 3.7.3 >.python-version</code> | 🐍 3.7.3 | the current python version in pyenv |

Other Zsh themes capable of displaying the same information either produce prompt lag or print prompt that doesn't reflect the current state of the system and then refresh it later. With Powerlevel10k you get fast prompt *and* up-to-date information.

FAQ: [Is it really fast?](#)

Powerlevel9k compatibility

Powerlevel10k understands all [Powerlevel9k](#) configuration parameters.

- ▶ Screen recording

[Migration](#) from Powerlevel9k to Powerlevel10k is a straightforward process. All your `POWERLEVEL9K` configuration parameters will still work. Prompt will look the same as before ([almost](#)) but it will be [much faster \(certainly\)](#).

FAQ:

- [I'm using Powerlevel9k with Oh My Zsh. How do I migrate?](#)
- [Does Powerlevel10k always render exactly the same prompt as Powerlevel9k given the same config?](#)
- [What is the relationship between Powerlevel9k and Powerlevel10k?](#)

Pure compatibility

Powerlevel10k can produce the same prompt as [Pure](#). Type `p10k configure` and select *Pure* style.

- ▶ Screen recording

You can still use Powerlevel10k features such as [transient prompt](#) or [instant prompt](#) when sporting Pure style.

To customize prompt, edit `~/.p10k.zsh`. Powerlevel10k doesn't recognize Pure configuration parameters, so you'll need to use `POWERLEVEL9K_COMMAND_EXECUTION_TIME_THRESHOLD=3` instead of `PURE_CMD_MAX_EXEC_TIME=3`, etc. All relevant parameters are in `~/.p10k.zsh`. This file has plenty of comments to help you navigate through it.

FAQ: [What is the best prompt style in the configuration wizard?](#)

Instant prompt

If your `~/.zshrc` loads many plugins, or perhaps just a few slow ones (for example, [pyenv](#) or [nvm](#)), you may have noticed that it takes some time for Zsh to start.

- ▶ Screen recording



Powerlevel10k can remove Zsh startup lag **even if it's not caused by a theme**.

- ▶ Screen recording

This feature is called *Instant Prompt*. You need to explicitly enable it through `p10k configure` or [manually](#). It does what it says on the tin -- prints prompt instantly upon Zsh startup allowing you to start typing while plugins are still loading.

Other themes *increase* Zsh startup lag -- some by a lot, others by just a little. Powerlevel10k *removes* it outright.

If you are curious about how *Instant Prompt* works, see [this section in zsh-bench](#).

FAQ: [How do I configure instant prompt?](#)

Show on command

The behavior of some commands depends on global environment. For example, `kubectl run ...` runs an image on the cluster defined by the current kubernetes context. If you frequently change context between "prod" and "testing", you might want to display the current context in Zsh prompt. If you do likewise for AWS, Azure and Google Cloud credentials, prompt will get pretty crowded.

Enter *Show On Command*. This feature makes prompt segments appear only when they are relevant to the command you are currently typing.

► Screen recording

Configs created by `p10k configure` enable show on command for several prompt segments by default. Here's the relevant parameter for kubernetes context:

```
# Show prompt segment "kubecontext" only when the command you are typing invokes one of these tools.
typeset -g POWERLEVEL9K_KUBECONTEXT_SHOW_ON_COMMAND='kubectl|helm|kubens'
```



To customize when different prompt segments are shown, open `~/.p10k.zsh`, search for `SHOW_ON_COMMAND` and either remove these parameters to display affected segments unconditionally, or change their values.

Transient prompt

When *Transient Prompt* is enabled through `p10k configure`, Powerlevel10k will trim down every prompt when accepting a command line.

► Screen recording

Transient prompt makes it much easier to copy-paste series of commands from the terminal scrollback.

Tip: If you enable transient prompt, take advantage of two-line prompt. You'll get the benefit of extra space for typing commands without the usual drawback of reduced scrollback density. Sparse prompt (with an empty line before prompt) also works great in combination with transient prompt.

Current directory that just works

The current working directory is perhaps the most important prompt segment. Powerlevel10k goes to great length to highlight its important parts and to truncate it with the least loss of information when horizontal space gets scarce.

► Screen recording

When the full directory doesn't fit, the leftmost segment gets truncated to its shortest unique prefix. In the screencast, `~/work` becomes `~/wo`. It couldn't be truncated to `~/w` because it would be ambiguous (there was `~/wireguard` when the session was recorded). The next segment `-- projects` turns into `p` as there was nothing else that started with `p` in `~/work/`.

Directory segments are shown in one of three colors:

- Truncated segments are bleak.
- Important segments are bright and never truncated. These include the first and the last segment, roots of Git repositories, etc.
- Regular segments (not truncated but can be) use in-between color.

Tip: If you copy-paste a truncated directory and hit `TAB`, it'll complete to the original.

Troubleshooting: [Directory is difficult to see in prompt when using Rainbow style](#).



Extremely customizable

Powerlevel10k can be configured to look like any other Zsh theme out there.

► Screen recording

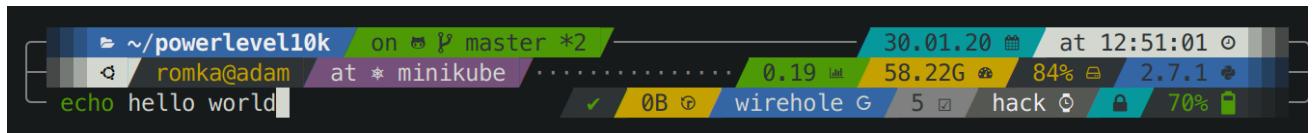
Pure, [Powerlevel9k](#) and [robbyrussell](#) emulations are built-in. To emulate the appearance of other themes, you'll need to write a

suitable configuration file. The best way to go about it is to run `p10k configure`, select the style that is the closest to your goal and then edit `~/.p10k.zsh`.

The full range of Powerlevel10k appearance spans from spartan:



To ridiculous extravagant:



Batteries included

Powerlevel10k comes with dozens of built-in high quality prompt segments that can display information from a variety of sources. When you run `p10k configure` and choose any style except `Pure`, many of these segments get enabled by default while others can be manually enabled by opening `~/.p10k.zsh` and uncommenting them. You can enable as many segments as you like. It won't slow down your prompt or Zsh startup.

| Segment | Meaning |
|------------------------|--|
| anaconda | virtual environment from conda |
| asdf | tool versions from asdf |
| aws | aws profile |
| aws_eb_env | aws elastic beanstalk environment |
| azure | azure account name |
| background_jobs | presence of background jobs |
| battery | internal battery state and charge level (yep, batteries <i>literally</i> included) |
| command_execution_time | duration (wall time) of the last command |
| context | user@hostname |
| cpu_arch | CPU architecture |
| dir | current working directory |
| direnv | direnv status |
| disk_usage | disk usage |
| dotnet_version | dotnet version |
| fvm | flutter environment from fvm |
| gcloud | google cloud cli account and project |
| goenv | go environment from goenv |
| google_app_cred | google application credentials |
| go_version | go version |
| haskell_stack | haskell version from stack |
| ip | IP address and bandwidth usage for a specified network interface |
| java_version | java version |
| jenv | java environment from jenv |
| kubecontext | current kubernetes context |
| laravel_version | laravel pho framework version |



| | View on GitHub |
|-----------------------|--|
| load | CPU load |
| luaenv | lua environment from luaenv |
| midnight_commander | midnight commander shell |
| nix_shell | nix shell indicator |
| nnn | nnn shell |
| lf | lf shell |
| chezmoi_shell | chezmoi shell |
| nodeenv | node.js environment from nodeenv |
| nodenv | node.js environment from nodenv |
| node_version | node.js version |
| nordvpn | nordvpn connection status |
| nvm | node.js environment from nvm |
| os_icon | your OS logo (apple for macOS, swirl for debian, etc.) |
| package | name@version from package.json |
| per_directory_history | Oh My Zsh per-directory-history local/global indicator |
| perlbrew | perl version from perlbrew |
| phenv | php environment from phenv |
| php_version | php version |
| plenv | perl environment from plenv |
| prompt_char | multi-functional prompt symbol; changes depending on vi mode: >, <, v, ▶ for insert, command, visual and replace mode respectively; turns red on error |
| proxy | system-wide http/https/ftp proxy |
| public_ip | public IP address |
| pyenv | python environment from pyenv |
| ram | free RAM |
| ranger | ranger shell |
| rbenv | ruby environment from rbenv |
| rust_version | rustc version |
| rvm | ruby environment from rvm |
| scalaenv | scala version from scalaenv |
| status | exit code of the last command |
| swap | used swap |
| taskwarrior | taskwarrior task count |
| terraform | terraform workspace |
| terraform_version | terraform version |
| time | current time |
| timewarrior | timewarrior tracking status |
| todo | todo items |
| toolbox | toolbox name |



| | |
|------------|---|
| vcs | Git repository status |
| vim_shell | vim shell (:sh) |
| virtualenv | python environment from venv |
| vi_mode | vi mode (you don't need this if you've enabled prompt_char) |
| vpn_ip | virtual private network indicator |
| wifi | WiFi speed |
| xplr | xplr shell |

Extensible

If there is no prompt segment that does what you need, implement your own. Powerlevel10k provides public API for defining segments that are as fast and as flexible as built-in ones.

- ▶ Screen recording

On Linux you can fetch current CPU temperature by reading `/sys/class/thermal/thermal_zone0/temp`. The screencast shows how to define a prompt segment to display this value. Once the segment is defined, you can use it like any other segment. All standard customization parameters will work for it out of the box.

Type `p10k help segment` for reference.

Note: If you modify `POWERLEVEL9K_*` parameters in an already initialized interactive shell (as opposed to editing `~/.p10k.zsh`), the changes might not be immediately effective. To apply the modifications, invoke `p10k reload`. Setting `POWERLEVEL9K_DISABLE_HOT_RELOAD=false` eliminates the necessity for `p10k reload` but results in a marginally slower prompt.

Tip: Prefix names of your own segments with `my_` to avoid clashes with future versions of Powerlevel10k.

Installation

- [Manual](#) ↗ choose this if confused or uncertain
- [Oh My Zsh](#)
- [Prezto](#)
- [Zim](#)
- [Antibody](#)
- [Antidote](#)
- [Antigen](#)
- [Zplug](#)
- [Zgen](#)
- [Zplugin](#)
- [Zinit](#)
- [Zi](#)
- [Zap](#)
- [Homebrew](#)
- [Arch Linux](#)
- [Alpine Linux](#)
- [Fig](#)

Manual

```
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ~/powerlevel10k
echo 'source ~/powerlevel10k/powerlevel10k.zsh-theme' >> ~/.zshrc
```



Users in China can use the official mirror on gitee.com for faster download.

中国用户可以使用 gitee.com 上的官方镜像加速下载。



```
git clone --depth=1 https://gitee.com/romkatv/powerlevel10k.git ~/powerlevel10k
echo 'source ~/powerlevel10k/powerlevel10k.zsh-theme' >> ~/.zshrc
```

This is the simplest kind of installation and it works even if you are using a plugin manager. Just make sure to disable the current theme in your plugin manager. See [troubleshooting](#) for help.

Oh My Zsh

1. Clone the repository:

```
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ${ZSH_CUSTOM:-$HOME/.oh-my-zsh/custom}/themes/powerlevel10k
```

Users in China can use the official mirror on gitee.com for faster download.

中国用户可以使用 gitee.com 上的官方镜像加速下载。

```
git clone --depth=1 https://gitee.com/romkatv/powerlevel10k.git ${ZSH_CUSTOM:-$HOME/.oh-my-zsh/custom}/themes/powerlevel10k
```

2. Set `ZSH_THEME="powerlevel10k/powerlevel10k"` in `~/.zshrc`.

Prezto

Add `zstyle :prezto:module:prompt theme powerlevel10k` to `~/.zpreztorc`.

Zim

Add `zmodule romkatv/powerlevel10k --use digit` to `~/.zimrc` and run `zimfw install`.

Antibody

Add `antibody bundle romkatv/powerlevel10k` to `~/.zshrc`.

Antidote

Add `romkatv/powerlevel10k` to `~/.zsh_plugins.txt`.

Antigen

Add `antigen theme romkatv/powerlevel10k` to `~/.zshrc`. Make sure you have `antigen apply` somewhere after it.

Zplug

Add `zplug romkatv/powerlevel10k, as:theme, depth:1` to `~/.zshrc`.

Zgen

Add `zgen load romkatv/powerlevel10k powerlevel10k` to `~/.zshrc`.

Zplugin

Add `zplugin ice depth:1; zplugin light romkatv/powerlevel10k` to `~/.zshrc`.

The use of `depth:1` `ice` is optional. Other types of `ice` are neither recommended nor officially supported by Powerlevel10k.

Zinit

Add `zinit ice depth:1; zinit light romkatv/powerlevel10k` to `~/.zshrc`.



The use of `depth:1` `ice` is optional. Other types of `ice` are neither recommended nor officially supported by Powerlevel10k.

Zi

Add `zi ice depth:1; zi light romkatv/powerlevel10k` to `~/.zshrc`.

The use of `depth:1` `ice` is optional. Other types of `ice` are neither recommended nor officially supported by Powerlevel10k.

The use of `zstyle` is optional. Other types of zle are neither recommended nor officially supported by Powerlevel10k.

Zap

Add `plug "romkatv/powerlevel10k"` to `~/.zshrc`.

Homebrew

```
brew install powerlevel10k
echo "source $(brew --prefix)/share/powerlevel10k/powerlevel10k.zsh-theme" >> ~/.zshrc
```



Arch Linux

```
yay -S --noconfirm zsh-theme-powerlevel10k-git
echo 'source /usr/share/zsh-theme-powerlevel10k/powerlevel10k.zsh-theme' >> ~/.zshrc
```



[zsh-theme-powerlevel10k-git](#) referenced above is the official Powerlevel10k package.

There is also [zsh-theme-powerlevel10k](#) package. Historically, [it has been breaking often and for extended periods of time](#). **Do not use it.**

Alpine Linux

```
apk add zsh zsh-theme-powerlevel10k
mkdir -p ~/.local/share/zsh/plugins
ln -s /usr/share/zsh/plugins/powerlevel10k ~/.local/share/zsh/plugins/
```



Fig

Follow the instructions on [this page](#).

Configuration

- [For new users](#)
- [For Powerlevel9k users](#)

For new users

On the first run, Powerlevel10k [configuration wizard](#) will ask you a few questions and configure your prompt. If it doesn't trigger automatically, type `p10k configure`. Configuration wizard creates `~/.p10k.zsh` based on your preferences. Additional prompt customization can be done by editing this file. It has plenty of comments to help you navigate through configuration options.

FAQ:

- [What is the best prompt style in the configuration wizard?](#)
- [What do different symbols in Git status mean?](#)
- [How do I change the format of Git status?](#)
- [How do I add username and/or hostname to prompt?](#)
- [How do I change prompt colors?](#)
- [Why some prompt segments appear and disappear as I'm typing?](#)

Troubleshooting:

- [Question mark in prompt.](#)
- [Icons, glyphs or powerline symbols don't render.](#)
- [Sub-pixel imperfections around powerline symbols.](#)
- [Directory is difficult to see in prompt when using Rainbow style.](#)



For Powerlevel9k users

If you've been using Powerlevel9k before, **do not remove the configuration options**. Powerlevel10k will pick them up and provide yo

with the same prompt UI you are used to. See [Powerlevel9k compatibility](#).

FAQ:

- [I'm using Powerlevel9k with Oh My Zsh. How do I migrate?](#)
- [What is the relationship between Powerlevel9k and Powerlevel10k?](#)
- [Does Powerlevel10k always render exactly the same prompt as Powerlevel9k given the same config?](#)

Troubleshooting: [Extra or missing spaces in prompt compared to Powerlevel9k.](#)

Fonts

Powerlevel10k doesn't require custom fonts but can take advantage of them if they are available. It works well with [Nerd Fonts](#), [Source Code Pro](#), [Font Awesome](#), [Powerline](#), and even the default system fonts. The full choice of style options is available only when using [Nerd Fonts](#).

👉 Recommended font: Meslo Nerd Font patched for Powerlevel10k. 👉

Meslo Nerd Font patched for Powerlevel10k

Gorgeous monospace font designed by Jim Lyles for Bitstream, customized by the same for Apple, further customized by André Berg and finally patched by yours truly with customized scripts originally developed by Ryan L McIntyre of Nerd Fonts. Contains all glyphs and symbols that Powerlevel10k may need. Battle-tested in dozens of different terminals on all major operating systems.

FAQ: [How was the recommended font created?](#)

Automatic font installation

If you are using iTerm2 or Termux, `p10k configure` can install the recommended font for you. Simply answer `Yes` when asked whether to install *Meslo Nerd Font*.

If you are using a different terminal, proceed with manual font installation. 👉

Manual font installation

1. Download these four ttf files:

- [MesloLGS NF Regular.ttf](#)
- [MesloLGS NF Bold.ttf](#)
- [MesloLGS NF Italic.ttf](#)
- [MesloLGS NF Bold Italic.ttf](#)

2. Double-click on each file and click "Install". This will make `MesloLGS NF` font available to all applications on your system.

3. Configure your terminal to use this font:

- **iTerm2:** Type `p10k configure` and answer `Yes` when asked whether to install *Meslo Nerd Font*. Alternatively, open *iTerm2* → *Preferences* → *Profiles* → *Text* and set *Font* to `MesloLGS NF`.
- **Apple Terminal:** Open *Terminal* → *Preferences* → *Profiles* → *Text*, click *Change* under *Font* and select `MesloLGS NF` family.
- **Hyper:** Open *Hyper* → *Edit* → *Preferences* and change the value of `fontFamily` under `module.exports.config` to `MesloLGS NF`.
- **Visual Studio Code:** Open *File* → *Preferences* → *Settings* (PC) or *Code* → *Preferences* → *Settings* (Mac), enter `terminal.integrated.fontFamily` in the search box at the top of *Settings* tab and set the value below to `MesloLGS NF`. Consult [this screenshot](#) to see how it should look like or see [this issue](#) for extra information.
- **GNOME Terminal** (the default Ubuntu terminal): Open *Terminal* → *Preferences* and click on the selected profile under *Profiles*. Check *Custom font* under *Text Appearance* and select `MesloLGS NF Regular`.
- **Konsole:** Open *Settings* → *Edit Current Profile* → *Appearance*, click *Select Font* and select `MesloLGS NF Regular`.
- **Tilix:** Open *Tilix* → *Preferences* and click on the selected profile under *Profiles*. Check *Custom font* under *Text Appearance* and select `MesloLGS NF Regular`.
- **Windows Console Host** (the old thing): Click the icon in the top left corner, then *Properties* → *Font* and set *Font* to `MesloLGS NF`.
- **Windows Terminal** by Microsoft (the new thing): Open *Settings* (`ctrl+,`), click either on the selected profile under *Profiles* or *Defaults*, click *Appearance* and set *Font face* to `MesloLGS NF`.
- **IntelliJ** (and other IDEs by Jet Brains): Open *IDE* → *Edit* → *Preferences* → *Editor* → *Color Scheme* → *Console Font*. Select *Use console font instead of the default* and set the font name to `MesloLGS NF`.
- **Termux:** Type `p10k configure` and answer `Yes` when asked whether to install *Meslo Nerd Font*.



- **Blink:** Type `config`, go to *Apearance*, tap *Add a new font*, tap *Open Gallery*, select `MesloLGS NF.css`, tap *import* and type `exit` in the home view to reload the font.
- **Taby (formerly Terminus):** Open *Settings* → *Apearance* and set *Font* to `MesloLGS NF`.
- **Terminator:** Open *Preferences* using the context menu. Under *Profiles* select the *General* tab (should be selected already), uncheck *Use the system fixed width font* (if not already) and select `MesloLGS NF Regular`. Exit the Preferences dialog by clicking *Close*.
- **Guake:** Right Click on an open terminal and open *Preferences*. Under *Apearance* tab, uncheck *Use the system fixed width font* (not already) and select `MesloLGS NF Regular`. Exit the Preferences dialog by clicking *Close*.
- **MobaXterm:** Open *Settings* → *Configuration* → *Terminal* → (under *Terminal look and feel*) and change *Font* to `MesloLGS NF`.
- **Asbrú Connection Manager:** Open *Preferences* → *Local Shell Options* → *Look and Feel*, enable *Use these personal options* and change *Font*: under *Terminal UI* to `MesloLGS NF Regular`. To change the font for the remote host connections, go to *Preferences* → *Terminal Options* → *Look and Feel* and change *Font*: under *Terminal UI* to `MesloLGS NF Regular`.
- **WSLtty:** Right click on an open terminal and then on *Options*. In the *Text* section, under *Font*, click *Select...* and set *Font* to `MesloLGS NF Regular`.
- **Yakuake:** Click \equiv → *Manage Profiles* → *New* → *Apearance*. Click *Choose* next to the *Font* dropdown, select `MesloLGS NF` and click *OK*. Click *OK* to save the profile. Select the new profile and click *Set as Default*.
- **Alacritty:** Create or open `~/.config/alacritty/alacritty.toml` and add the following section to it:

```
[font.normal]
family = "MesloLGS NF"
```



- **foot:** Create or open `~/.config/foot/foot.ini` and add the following section to it:

```
font=MesloLGS NF:size=12
```



- **kitty:** Create or open `~/.config/kitty/kitty.conf` and add the following line to it:

```
font_family MesloLGS NF
```



Restart kitty by closing all sessions and opening a new session.

- **puTTY:** Set *Window* → *Apearance* → *Font* to `MesloLGS NF`. Requires puTTY version ≥ 0.75 .
- **WezTerm:** Create or open `$HOME/.config/wezterm/wezterm.lua` and add the following:

```
local wezterm = require 'wezterm';
return {
    font = wezterm.font("MesloLGS NF"),
}
```



If the file already exists, only add the line with the font to the existing return. Also add the first line if it is not already present

- **urxvt:** Create or open `~/.Xresources` and add the following line to it:

```
URxvt.font: xft:MesloLGS NF:size=11
```



You can adjust the font size to your preference. After changing the config run `xrdb ~/.Xresources` to reload it. The new config is applied to all new terminals.

- **xterm:** Create or open `~/.Xresources` and add the following line to it:

```
xterm*faceName: MesloLGS NF
```



After changing the config run `xrdb ~/.Xresources` to reload it. The new config is applied to all new terminals.

- **Zed:** Open `~/.config/zed/settings.json` and set `terminal.font_family` to `"MesloLGS NF"`.

```
{
    "terminal": {
        "font_family": "MesloLGS NF"
    },
    // Other settings.
}
```



- **Crostini (Linux on Chrome OS):** Open `chrome-untrusted://terminal/html/nassh_preferences_editor.html`, set *Text font family* to `'MesloLGS NF'` (including the quotes) and *Custom CSS (inline text)* to the following:

```
@font-face {  
    font-family: "MesloLGS NF";  
    src: url("https://raw.githubusercontent.com/romkatv/powerlevel10k-media/master/MesloLGS%20NF%20Regular.ttf")  
    font-weight: normal;  
    font-style: normal;  
}  
@font-face {  
    font-family: "MesloLGS NF";  
    src: url("https://raw.githubusercontent.com/romkatv/powerlevel10k-media/master/MesloLGS%20NF%20Bold.ttf")  
    font-weight: bold;  
    font-style: normal;  
}  
@font-face {  
    font-family: "MesloLGS NF";  
    src: url("https://raw.githubusercontent.com/romkatv/powerlevel10k-media/master/MesloLGS%20NF%20Italic.ttf")  
    font-weight: normal;  
    font-style: italic;  
}  
@font-face {  
    font-family: "MesloLGS NF";  
    src: url("https://raw.githubusercontent.com/romkatv/powerlevel10k-media/master/MesloLGS%20NF%20Bold%20Ita  
    font-weight: bold;  
    font-style: italic;  
}
```

CAVEAT: If you open the normal terminal preferences these settings will be overwritten.

4. Run `p10k configure` to generate a new `~/.p10k.zsh`. The old config may work incorrectly with the new font.

Using a different terminal and know how to set the font for it? Share your knowledge by sending a PR to expand the list!

Try it in Docker

Try Powerlevel10k in Docker. You can safely make any changes to the file system while trying out the theme. Once you exit Zsh, the container is deleted.

```
docker run -e TERM -e COLORTERM -e LC_ALL=C.UTF-8 -it --rm alpine sh -uec '  
apk add git zsh nano vim  
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ~/powerlevel10k  
echo "source ~/powerlevel10k/powerlevel10k.zsh-theme" >>~/.zshrc  
cd ~/powerlevel10k  
exec zsh'
```

Tip: Install [the recommended font](#) before running the Docker command to get access to all prompt styles.

Tip: Run `p10k configure` while in Docker to try a different prompt style.

License

Powerlevel10k is released under the [MIT license](#).

FAQ

- [How do I update Powerlevel10k?](#)
- [How do I uninstall Powerlevel10k?](#)
- [How do I install Powerlevel10k on a machine without Internet access?](#)
- [Where can I ask for help and report bugs?](#)
- [Which aspects of shell and terminal does Powerlevel10k affect?](#)
- [I'm using Powerlevel9k with Oh My Zsh. How do I migrate?](#)
- [Is it really fast?](#)
- [How do I configure instant prompt?](#)
- [How do I initialize direnv when using instant prompt?](#)
- [How do I export GPG_TTY when using instant prompt?](#)
- [What do different symbols in Git status mean?](#)



- [What do different symbols in Git status mean?](#)
- [How do I change the format of Git status?](#)
- [Why is Git status from `\$HOME/.git` not displayed in prompt?](#)
- [Why does Git status sometimes appear grey and then gets colored after a short period of time?](#)
- [How do I add username and/or hostname to prompt?](#)
- [Why some prompt segments appear and disappear as I'm typing?](#)
- [How do I change prompt colors?](#)
- [Why does Powerlevel10k spawn extra processes?](#)
- [Are there configuration options that make Powerlevel10k slow?](#)
- [Is Powerlevel10k fast to load?](#)
- [What is the relationship between Powerlevel9k and Powerlevel10k?](#)
- [Does Powerlevel10k always render exactly the same prompt as Powerlevel9k given the same config?](#)
- [What is the best prompt style in the configuration wizard?](#)
- [How to make Powerlevel10k look like robyrussell Oh My Zsh theme?](#)
- [Can prompts for completed commands display error status for those commands instead of the commands preceding them?](#)
- [What is the minimum supported Zsh version?](#)
- [How were these screenshots and animated gifs created?](#)
- [How was the recommended font created?](#)
- [How to package Powerlevel10k for distribution?](#)

How do I update Powerlevel10k?

The command to update Powerlevel10k depends on how it was installed.

| Installation | Update command |
|------------------------------|--|
| Manual | <code>git -C ~/powerlevel10k pull</code> |
| Oh My Zsh | <code>git -C \${ZSH_CUSTOM:-\$HOME/.oh-my-zsh/custom}/themes/powerlevel10k pull</code> |
| Prezto | <code>zprezto-update</code> |
| Zim | <code>zimfw update</code> |
| Antigen | <code>antigen update</code> |
| Antidote | <code>antidote update</code> |
| Zplug | <code>zplug update</code> |
| Zgen | <code>zgen update</code> |
| Zplugin | <code>zplugin update</code> |
| Zinit | <code>zinit update</code> |
| Zi | <code>zi update</code> |
| Zap | <code>zap update</code> |
| Homebrew | <code>brew update && brew upgrade</code> |
| Arch Linux | <code>yay -S --noconfirm zsh-theme-powerlevel10k-git</code> |
| Alpine Linux | <code>apk update && apk upgrade</code> |

IMPORTANT: Restart Zsh after updating Powerlevel10k. [Do not use `source ~/.zshrc`.](#)

How do I uninstall Powerlevel10k?

1. Remove all references to "p10k" from `~/.zshrc`. You might have this snippet at the top:

```
if [[ -r "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({(%):-%n}.zsh" ]]; then
    source "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({(%):-%n}.zsh"
fi
```



And this at the bottom:

```
[[ ! -f ~/.p10k.zsh ]] || source ~/.p10k.zsh
```



These are added by the [configuration wizard](#). Remove them.

2. Remove all references to "powerlevel10k" from `~/.zshrc`, `~/.zpreztorc` and `~/.zimrc` (some of these files may be missing -- this is normal). These references have been added manually by yourself when installing Powerlevel10k. Refer to the [installation instructions](#) if you need a reminder.
3. Verify that all references to "p10k" and "powerlevel10k" are gone from `~/.zshrc`, `~/.zpreztorc` and `~/.zimrc`.

```
grep -E 'p10k|powerlevel10k' ~/.zshrc ~/.zpreztorc ~/.zimrc 2>/dev/null
```



If this command produces output, there are still references to "p10k" or "powerlevel10k". You need to remove them.

4. Delete Powerlevel10k configuration file. This file is created by the [configuration wizard](#) and may contain manual edits by yourself

```
rm -f ~/.p10k.zsh
```



5. Delete Powerlevel10k source files. These files have been downloaded when you've installed Powerlevel10k. The command to delete them depends on which installation method you'd chosen. Refer to the [installation instructions](#) if you need a reminder.

| Installation | Uninstall command |
|------------------------------|--|
| Manual | <code>rm -rf ~/powerlevel10k</code> |
| Oh My Zsh | <code>rm -rf -- \${ZSH_CUSTOM:-\$HOME/.oh-my-zsh/custom}/themes/powerlevel10k</code> |
| Prezto | n/a |
| Zim | <code>zimfw uninstall</code> |
| Antigen | <code>antigen purge romkatv/powerlevel10k</code> |
| Antidote | <code>antidote purge romkatv/powerlevel10k</code> |
| Zplug | <code>zplug clean</code> |
| Zgen | <code>zgen reset</code> |
| Zplugin | <code>zplugin delete romkatv/powerlevel10k</code> |
| Zinit | <code>zinit delete romkatv/powerlevel10k</code> |
| Zi | <code>zi delete romkatv/powerlevel10k</code> |
| Zap | <code>zsh -ic 'zap clean'</code> |
| Homebrew | <code>brew uninstall powerlevel10k</code> |
| Arch Linux | <code>yay -R --noconfirm zsh-theme-powerlevel10k-git</code> |
| Alpine Linux | <code>apk del zsh-theme-powerlevel10k</code> |

6. Restart Zsh. [Do not use `source ~/.zshrc`](#).

7. Delete Powerlevel10k cache files.

```
rm -rf -- "${XDG_CACHE_HOME:-$HOME/.cache}"/p10k-*(*N) "${XDG_CACHE_HOME:-$HOME/.cache}"/gitstatus
```



How do I install Powerlevel10k on a machine without Internet access?

1. Run this command on the machine without Internet access:

```
uname -sm | tr '[A-Z]' '[a-z]'
```



2. Run these commands on a machine connected to the Internet after replacing the value of `target_uname` with the output of the previous command:

```
target_uname="replace this with the output of the previous command"
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ~/powerlevel10k
GITSTATUS_CACHE_DIR="$HOME"/powerlevel10k/gitstatus/usrbin ~/powerlevel10k/gitstatus/install -f -s "${target_unam
```



3. Copy `~/powerlevel10k` from the machine connected to the Internet to the one without Internet access.

4. Add `source ~/powerlevel10k/powerlevel10k.zsh-theme` to `~/.zshrc` on the machine without Internet access:

```
echo 'source ~/powerlevel10k/powerlevel10k.zsh-theme' >> ~/.zshrc
```



5. If `~/.zshrc` on the machine without Internet access sets `ZSH_THEME`, remove that line.

```
sed -i.bak '/^ZSH_THEME=/d' ~/.zshrc
```



To update, remove `~/powerlevel10k` on both machines and repeat steps 1-3.

Where can I ask for help and report bugs?

The best way to ask for help and to report bugs is to [open an issue](#).

[Gitter](#) is another option.

If all else fails, email roman.perepelitsa@gmail.com.

If necessary, encrypt your communication with [this PGP key](#).

Which aspects of shell and terminal does Powerlevel10k affect?

Powerlevel10k defines prompt and nothing else. It sets [prompt-related options](#), and parameters `PS1` and `RPS1`.

```
> ls
docs .gitattributes LICENSE zsh-defer
.git .gitignore README.md zsh-defer.plugin.zsh

~/dotfiles/zsh-defer @e220a3ac ..... 08:29:24
> echo hello world
```

Everything within the highlighted areas on the screenshot is produced by Powerlevel10k. Powerlevel10k has no control over the terminal content or colors outside these areas.

Powerlevel10k does not affect:

- Terminal window/tab title.
- Colors used by `ls`.
- The behavior of `git` command.
- The content and style of `Tab` completions.
- Command line colors (syntax highlighting, autosuggestions, etc.).
- Key bindings.
- Aliases.
- Prompt parameters other than `PS1` and `RPS1`.
- Zsh options other than those [related to prompt](#).
- The set of available commands. Powerlevel10k does not install any new commands with the only exception of `p10k`.



I'm using Powerlevel9k with Oh My Zsh. How do I migrate?

1. Run this command:

```
# Add powerlevel10k to the list of Oh My Zsh themes.
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git $ZSH_CUSTOM/themes/powerlevel10k
# Replace ZSH_THEME="powerlevel9k/powerlevel9k" with ZSH_THEME="powerlevel10k/powerlevel10k".
sed -i.bak 's/powerlevel9k/powerlevel10k/g' ~/.zshrc
# Restart Zsh.
exec zsh
```

2. Optional but highly recommended:

- Install [the recommended font](#).
- Type `p10k configure` and choose your favorite prompt style.

Related:

- [Powerlevel9k compatibility.](#)
- [Does Powerlevel10k always render exactly the same prompt as Powerlevel9k given the same config?](#)
- [Extra or missing spaces in prompt compared to Powerlevel9k.](#)
- [Configuration wizard.](#)

Is it really fast?

Yes. See [zsh-bench](#) or a direct comparison with [Powerlevel9k](#) and [Spaceship](#).

How do I configure instant prompt?

See [instant prompt](#) to learn about instant prompt. This section explains how you can enable and configure it and lists caveats that you should be aware of.

Instant prompt can be enabled either through `p10k configure` or by manually adding the following code snippet at the top of `~/.zshrc`:

```
# Enable Powerlevel10k instant prompt. Should stay close to the top of ~/.zshrc.
# Initialization code that may require console input (password prompts, [y/n]
# confirmations, etc.) must go above this block; everything else may go below.
if [[ -r "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-%n}.zsh" ]]; then
    source "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-%n}.zsh"
fi
```

It's important that you copy the lines verbatim. Don't replace `source` with something else, don't call `zcompile`, don't redirect output etc.

When instant prompt is enabled, for the duration of Zsh initialization standard input is redirected to `/dev/null` and standard output with standard error are redirected to a temporary file. Once Zsh is fully initialized, standard file descriptors are restored and the content of the temporary file is printed out.

When using instant prompt, you should carefully check any output that appears on Zsh startup as it may indicate that initialization has been altered, or perhaps even broken, by instant prompt. Initialization code that may require console input, such as asking for a keyring password or for a `[y/n]` confirmation, must be moved above the instant prompt preamble in `~/.zshrc`. Initialization code that merely prints to console but never reads from it will work correctly with instant prompt, although output that normally has colors may appear uncolored. You can either leave it be, suppress the output, or move it above the instant prompt preamble.

Here's an example of `~/.zshrc` that breaks when instant prompt is enabled:

```
if [[ -r "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-%n}.zsh" ]]; then
    source "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-%n}.zsh"
fi

keychain id_rsa --agents ssh # asks for password
chatty-script # spams to stdout even when everything is fine
# ...
```

Fixed version:

```
keychain id_rsa --agents ssh # moved before instant prompt
```

```
# OK to perform console I/O before this point.
if [[ -r "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-n}.zsh" ]]; then
    source "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-n}.zsh"
fi
# From this point on, until zsh is fully initialized, console input won't work and
# console output may appear uncolored.

chatty-script >/dev/null      # spam output suppressed
# ...
```

If `POWERLEVEL9K_INSTANT_PROMPT` is unset or set to `verbose`, Powerlevel10k will print a warning when it detects console output during initialization to bring attention to potential issues. You can silence this warning (without suppressing console output) with `POWERLEVEL9K_INSTANT_PROMPT=quiet`. This is recommended if some initialization code in `~/.zshrc` prints to console and it's infeasible to move it above the instant prompt preamble or to suppress its output. You can completely disable instant prompt with `POWERLEVEL9K_INSTANT_PROMPT=off`. Do this if instant prompt breaks Zsh initialization and you don't know how to fix it.

The value of `POWERLEVEL9K_INSTANT_PROMPT` can be changed by running `p10k configure` and selecting the appropriate option on the *Instant Prompt* screen. Alternatively, you can search for `POWERLEVEL9K_INSTANT_PROMPT` in the existing `~/.p10k.zsh` and change its value there.

Note: Instant prompt requires Zsh >= 5.4. It's OK to enable it even when using an older version of Zsh but it won't do anything.

FAQ:

- [How do I initialize direnv when using instant prompt?](#)
- [How do I export GPG_TTY when using instant prompt?](#)

How do I initialize direnv when using instant prompt?

If you've enabled [instant prompt](#), you should have these lines at the top of `~/.zshrc`:

```
if [[ -r "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-n}.zsh" ]]; then
    source "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-n}.zsh"
fi
```

To initialize direnv you need to add one line above that block and one line below it.

```
(( ${+commands[direnv]} ) ) && emulate zsh -c "$(direnv export zsh)"

if [[ -r "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-n}.zsh" ]]; then
    source "${XDG_CACHE_HOME:-$HOME/.cache}/p10k-instant-prompt-$({%):-n}.zsh"
fi

(( ${+commands[direnv]} ) ) && emulate zsh -c "$(direnv hook zsh)"
```

Related: [How do I export GPG_TTY when using instant prompt?](#)

How do I export GPG_TTY when using instant prompt?

You can export `GPG_TTY` like this anywhere in `~/.zshrc`:

```
export GPG_TTY=$TTY
```

This works whether you are using [instant prompt](#) or not. It works even if you aren't using powerlevel10k. As an extra bonus, it's much faster than the commonly used `export GPG_TTY=$(tty)`.

Related: [How do I initialize direnv when using instant prompt?](#)

What do different symbols in Git status mean?

When using Lean, Classic or Rainbow style, Git status may look like this:

```
feature:master wip ↗42↑42 ←42→42 *42 merge ~42 +42 !42 ?42
```

| Symbol | Meaning | Source |
|---------|---|--|
| feature | current branch; replaced with #tag or @commit if not on a branch | git status --ignore-submodules=dirty |
| master | remote tracking branch; only shown if different from local branch | git rev-parse --abbrev-ref --symbolic-full-name @{upstream} |
| wip | the latest commit's summary contains "wip" or "WIP" | git show --pretty=%s --no-patch HEAD |
| = | up to date with the remote (neither ahead nor behind) | git rev-list --count HEAD...@{upstream} |
| ↓42 | this many commits behind the remote | git rev-list --right-only --count HEAD...@{upstream} |
| ↑42 | this many commits ahead of the remote | git rev-list --left-only --count HEAD...@{upstream} |
| ↔42 | this many commits behind the push remote | git rev-list --right-only --count HEAD...@{push} |
| →42 | this many commits ahead of the push remote | git rev-list --left-only --count HEAD...@{push} |
| *42 | this many stashes | git stash list |
| merge | repository state | git status --ignore-submodules=dirty |
| ~42 | this many merge conflicts | git status --ignore-submodules=dirty |
| +42 | this many staged changes | git status --ignore-submodules=dirty |
| !42 | this many unstaged changes | git status --ignore-submodules=dirty |
| ?42 | this many untracked files | git status --ignore-submodules=dirty |
| — | the number of staged, unstaged or untracked files is unknown | echo \$POWERLEVEL9K_VCS_MAX_INDEX_SIZE_DIRTY or git config --get bash.showDirtyState |

Related: [How do I change the format of Git status?](#)

How do I change the format of Git status?

To change the format of Git status, open `~/.p10k.zsh`, search for `my_git_formatter` and edit its source code.

Related: [What do different symbols in Git status mean?](#)

Why is Git status from `$HOME/.git` not displayed in prompt?

When using Lean, Classic or Rainbow style, `~/.p10k.zsh` contains the following parameter:

```
# Don't show Git status in prompt for repositories whose workdir matches this pattern.
# For example, if set to '~', the Git repository at $HOME/.git will be ignored.
# Multiple patterns can be combined with '|': '~(/foo)|/bar/baz/*'.
typeset -g POWERLEVEL9K_VCS_DISABLED_WORKDIR_PATTERN='~'
```



To see Git status for `$HOME/.git` in prompt, open `~/.p10k.zsh` and remove `POWERLEVEL9K_VCS_DISABLED_WORKDIR_PATTERN`.

Why does Git status sometimes appear grey and then gets colored after a short period of time?

tl;dr: When Git status in prompt is greyed out, it means Powerlevel10k is currently computing up-to-date Git status in the background. Prompt will get automatically refreshed when this computation completes.



When your current directory is within a Git repository, Powerlevel10k computes up-to-date Git status after every command. If the repository is large, or the machine is slow, this computation can take quite a bit of time. If it takes longer than 10 milliseconds (configurable via `POWERLEVEL9K_VCS_MAX_SYNC_LATENCY_SECONDS`), Powerlevel10k displays the last known Git status in grey and continues to compute up-to-date Git status in the background. When the computation completes, Powerlevel10k refreshes prompt with new information, this time with colored Git status.

When using *Rainbow* style, Git status is displayed as black on grey while it's still being computed. Depending on the terminal color palette, this may be difficult to read. In this case you might want to change the background color to something lighter for more contrast. To do that, open `~/.p10k.zsh`, search for `POWERLEVEL9K_VCS_LOADING_BACKGROUND`, uncomment it if it's commented out, and change the value.

```
typeset -g POWERLEVEL9K_VCS_LOADING_BACKGROUND=244
```



Type `source ~/.p10k.zsh` to apply your changes to the current Zsh session.

Related: [How do I change prompt colors?](#)

How do I add username and/or hostname to prompt?

When using Lean, Classic or Rainbow style, prompt shows `username@hostname` when you are logged in as root or via SSH. There is little value in showing `username` or `hostname` when you are logged in to your local machine as a normal user. So the absence of `username@hostname` in your prompt is an indication that you are working locally and that you aren't root. You can change it, however.

Open `~/.p10k.zsh`. Close to the top you can see the most important parameters that define which segments are shown in your prompt. All generally useful prompt segments are listed in there. Some of them are enabled, others are commented out. One of them is of interest to you.

```
typeset -g POWERLEVEL9K_RIGHT_PROMPT_ELEMENTS=(  
    ...  
    context # user@hostname  
    ...  
)
```



Search for `context` to find the section in the config that lists parameters specific to this prompt segment. You should see the following lines:

```
# Don't show context unless running with privileges or in SSH.  
# Tip: Remove the next line to always show context.  
typeset -g POWERLEVEL9K_CONTEXT_{DEFAULT,SUDO}_{CONTENT,VISUAL_IDENTIFIER}_EXPANSION=
```



If you follow the tip and remove (or comment out) the last line, you'll always see `username@hostname` in prompt. You can change the format to just `username`, or change the color, by adjusting the values of parameters nearby. There are plenty of comments to help you navigate.

You can also move `context` to a different position in `POWERLEVEL9K_RIGHT_PROMPT_ELEMENTS` or even to `POWERLEVEL9K_LEFT_PROMPT_ELEMENTS`.

Why some prompt segments appear and disappear as I'm typing?

Prompt segments can be configured to be shown only when the current command you are typing invokes a relevant tool.

```
# Show prompt segment "kubectx" only when the command you are typing invokes  
# invokes kubectl, helm, or kubens.  
typeset -g POWERLEVEL9K_KUBECONTEXT_SHOW_ON_COMMAND='kubectl|helm|kubens'
```



Configs created by `p10k configure` may contain parameters of this kind. To customize when different prompt segments are shown, open `~/.p10k.zsh`, search for `SHOW_ON_COMMAND` and either remove these parameters or change their values.

You can also define a function in `~/.zshrc` to toggle the display of a prompt segment between *always* and *on command*. This is similar to `kubeon` / `kubeoff` from [kube-ps1](#).

```
function kube-toggle() {  
    if (( ${+POWERLEVEL9K_KUBECONTEXT_SHOW_ON_COMMAND} )); then  
        unset POWERLEVEL9K_KUBECONTEXT_SHOW_ON_COMMAND  
    else  
        POWERLEVEL9K_KUBECONTEXT_SHOW_ON_COMMAND='kubectl|helm|kubens'  
    fi  
    p10k reload  
    if zle; then
```



```

    zle push-input
    zle accept-line
}

```

Invoke this function by typing `kube-toggle`. You can also bind it to a key by adding two more lines to `~/.zshrc`:

```

zle -N kube-toggle
bindkey '^]' kube-toggle # ctrl-] to toggle kubecontext in powerlevel10k prompt

```



How do I change prompt colors?

You can either [change the color palette used by your terminal](#) or [set colors through Powerlevel10k configuration parameters](#).

Change the color palette used by your terminal

How exactly you change the terminal color palette (a.k.a. color scheme, or theme) depends on the kind of terminal you are using. Look around in terminal's settings/preferences or consult documentation.

When you change the terminal color palette, it usually affects only the first 16 colors, numbered from 0 to 15. In order to see any effect on Powerlevel10k prompt, you need to use prompt style that utilizes these low-numbered colors. Type `p10k configure` and select *Rainbow*, *Lean* → *8 colors* or *Pure* → *Original*. Other styles use higher-numbered colors, so they look the same in any terminal color palette.

Set colors through Powerlevel10k configuration parameters

Open `~/.p10k.zsh`, search for "color", "foreground" and "background" and change values of appropriate parameters. For example, here's how you can set the foreground of `time` prompt segment to bright red:

```
typeset -g POWERLEVEL9K_TIME_FOREGROUND=160
```



Colors are specified using numbers from 0 to 255. Colors from 0 to 15 look differently in different terminals. Many terminals also support customization of these colors through color palettes (a.k.a. color schemes, or themes). Colors from 16 to 255 always look the same.

Type `source ~/.p10k.zsh` to apply your changes to the current Zsh session.

To see how different numbered colors look in your terminal, run the following command:

```
for i in {0..255}; do print -Pn "%K{$i} %k%F{$i}${(l:3::0:)i}%f ${${{(M)$((i%6)):#3}}:+$'\n'}; done
```



If your terminal supports truecolor, you can use 24-bit colors in the `#RRGGBB` format in addition to the numbered colors.

```
typeset -g POWERLEVEL9K_TIME_FOREGROUND='#FF0000'
```



Related:

- [Directory is difficult to see in prompt when using Rainbow style.](#)

Why does Powerlevel10k spawn extra processes?

Powerlevel10k uses [gitstatus](#) as the backend behind `vcs` prompt; gitstatus spawns `gitstatusd` and `zsh`. See [gitstatus](#) for details. Powerlevel10k may also spawn `zsh` to perform computation without blocking prompt. To avoid security hazard, these background processes aren't shared by different interactive shells. They terminate automatically when the parent `zsh` process terminates or run `exec(3)`.



Are there configuration options that make Powerlevel10k slow?

No, Powerlevel10k is always fast, with any configuration you throw at it. If you have noticeable prompt latency when using Powerlevel10k, please [open an issue](#).

Is Powerlevel10k fast to load?

Yes. See [zsh-bench](#).

What is the relationship between Powerlevel9k and Powerlevel10k?

Powerlevel10k was forked from Powerlevel9k in March 2019 after a week-long discussion in [powerlevel9k#1170](#). Powerlevel9k was already a mature project with a large user base and a release cycle measured in months. Powerlevel10k was spun off to iterate on performance improvements and new features at much higher pace.

Powerlevel9k and Powerlevel10k are independent projects. When using one, you shouldn't install the other. Issues should be filed against the project that you actually use. There are no individuals that have commit rights in both repositories. All bug fixes and new features committed to Powerlevel9k repository get ported to Powerlevel10k.

Over time, virtually all code in Powerlevel10k has been rewritten. There is currently no meaningful overlap between the implementations of Powerlevel9k and Powerlevel10k.

Powerlevel10k is committed to maintaining backward compatibility with all configs indefinitely. This commitment covers all configuration parameters recognized by Powerlevel9k (see [Powerlevel9k compatibility](#)) and additional parameters that only Powerlevel10k understands. Names of all parameters in Powerlevel10k start with `POWERLEVEL9K_` for consistency.

Does Powerlevel10k always render exactly the same prompt as Powerlevel9k given the same config?

Almost. There are a few differences.

- By default only `git vcs` backend is enabled in Powerlevel10k. If you need `svn` and `hg`, add them to `POWERLEVEL9K_VCS_BACKENDS`. These backends aren't yet optimized in Powerlevel10k, so enabling them will make prompt *very slow*.
- Powerlevel10k doesn't support `POWERLEVEL9K_VCS_SHOW_SUBMODULE_DIRTY=true`.
- Powerlevel10k strives to be bug-compatible with Powerlevel9k but not when it comes to egregious bugs. If you accidentally rely on these bugs, your prompt will differ between Powerlevel9k and Powerlevel10k. Some examples:
 - Powerlevel9k ignores some options that are set after the theme is sourced while Powerlevel10k respects all options. If you see different icons in Powerlevel9k and Powerlevel10k, you've probably defined `POWERLEVEL9K_MODE` before sourcing the theme. This parameter gets ignored by Powerlevel9k but honored by Powerlevel10k. If you want your prompt to look in Powerlevel10k the same as in Powerlevel9k, remove `POWERLEVEL9K_MODE`.
 - Powerlevel9k doesn't respect `ZLE_RPROMPT_INDENT`. As a result, right prompt in Powerlevel10k can have an extra space at the end compared to Powerlevel9k. Set `ZLE_RPROMPT_INDENT=0` if you don't want that space. More details in [troubleshooting](#).
 - Powerlevel9k has inconsistent spacing around icons. This was fixed in Powerlevel10k. Set `POWERLEVEL9K_LEGACY_ICON_SPACING=true` to get the same spacing as in Powerlevel9k. More details in [troubleshooting](#).
 - There are dozens more bugs in Powerlevel9k that don't exist in Powerlevel10k.

If you notice any other changes in prompt appearance when switching from Powerlevel9k to Powerlevel10k, please [open an issue](#).

What is the best prompt style in the configuration wizard?

There are as many opinions on what constitutes the best prompt as there are people. It mostly comes down to personal preference. There are, however, a few hidden implications of different choices.

Pure style is an exact replication of [Pure Zsh theme](#). It exists to ease the migration for users of this theme. Unless you are one of them choose Lean style over Pure.

If you want to confine prompt colors to the selected terminal color palette (say, *Solarized Dark*), use *Rainbow*, *Lean → 8 colors* or *Pure - Original*. Other styles use fixed colors and thus look the same in any terminal color palette.

All styles except Pure have an option to use *ASCII* charset. Prompt will look less pretty but will render correctly with all fonts and in all locales.

If you enable transient prompt, take advantage of two-line prompt. You'll get the benefit of extra space for typing commands without the usual drawback of reduced scrollback density. Having all commands start from the same offset is also nice.



Similarly, if you enable transient prompt, sparse prompt (with an empty line before prompt) is a great choice.

If you are using vi keymap, choose prompt with `prompt_char` in it (shown as green `>` in the wizard). This symbol changes depending on vi mode: `>`, `<`, `v`, `▶` for insert, command, visual and replace mode respectively. When a command fails, the symbol turns red. *Lean* style always has `prompt_char` in it. *Rainbow* and *Classic* styles have it only in the two-line configuration without left frame.

If you value horizontal space or prefer minimalist aesthetics:

- Use a monospace font, such as [the recommended font](#). Non-monospace fonts require extra space after icons that are larger than a single column.
- Use Lean style. Compared to Classic and Rainbow, it saves two characters per prompt segment.
- Disable *current time* and *frame*.
- Use *few icons*. The extra icons enabled by the *many icons* option primarily serve decorative function. Informative icons, such as background job indicator, will be shown either way.

Note: You can run configuration wizard as many times as you like. Type `p10k configure` to try new prompt style.

How to make Powerlevel10k look like robbyrussell Oh My Zsh theme?

Use [this config](#).

You can either download it, save as `~/.p10k.zsh` and `source ~/.p10k.zsh` from `~/.zshrc`, or source `p10k-robbyrussell.zsh` direct from your cloned `powerlevel10k` repository.

Can prompts for completed commands display error status for *those* commands instead of the commands preceding them?

No. When you hit *ENTER* and the command you've typed starts running, its error status isn't yet known, so it cannot be shown in prompt. When the command completes, the error status gets known but it's no longer possible to update prompt for *that* command. This is why the error status for every command is reflected in the *next* prompt.

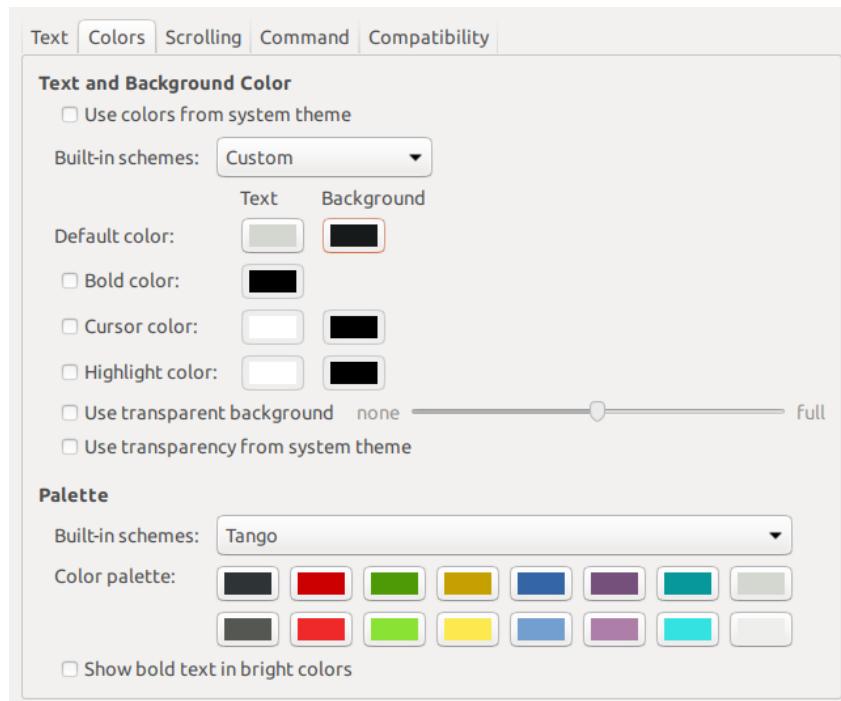
For details, see [this post on /r/zsh](#).

What is the minimum supported Zsh version?

Zsh 5.3 or newer should work. Fast startup requires Zsh ≥ 5.4 .

How were these screenshots and animated gifs created?

All screenshots and animated gifs were recorded in GNOME Terminal with [the recommended font](#) and Tango Dark color palette with custom background color (`#171A1B` instead of `#2E3436` -- twice as dark).



Syntax highlighting, where present, was provided by [zsh-syntax-highlighting](#).

How was the recommended font created?

[The recommended font](#) is the product of many individuals. Its origin is *Bitstream Vera Sans Mono*, which has given birth to *Menlo*, which

in turn has spawned *Meslo*. Finally, extra glyphs have been added to *Meslo* with scripts forked from Nerd Fonts. The final font is released under the terms of [Apache License](#).

MesloLGS NF font can be recreated with the following command (requires `git` and `docker`):

```
git clone --depth=1 https://github.com/romkatv/nerd-fonts.git
cd nerd-fonts
./build 'Meslo/S/*'
```

If everything goes well, four `ttf` files will appear in `./out`.

How to package Powerlevel10k for distribution?

It's currently neither easy nor recommended to package and distribute Powerlevel10k. There are no instructions you can follow that would allow you to easily update your package when new versions of Powerlevel10k are released. This may change in the future but not soon.

Troubleshooting

- [\[oh-my-zsh\] theme 'powerlevel10k/powerlevel10k' not found](#)
- [Question mark in prompt](#)
- [Icons, glyphs or powerline symbols don't render](#)
- [Sub-pixel imperfections around powerline symbols](#)
- [Error: character not in range](#)
- [Cursor is in the wrong place](#)
- [Prompt wrapping around in a weird way](#)
- [Right prompt is in the wrong place](#)
- [Configuration wizard runs automatically every time Zsh is started](#)
- [Some prompt styles are missing from the configuration wizard](#)
- [Cannot install the recommended font](#)
- [Extra or missing spaces in prompt compared to Powerlevel9k](#)
 - [Extra space without background on the right side of right prompt](#)
 - [Extra or missing spaces around icons](#)
- [Weird things happen after typing `_source ~/.zshrc`](#)
- [Transient prompt stops working after some time](#)
- [Cannot make Powerlevel10k work with my plugin manager](#)
- [Directory is difficult to see in prompt when using Rainbow style](#)
- [Horrific mess when resizing terminal window](#)
- [Icons cut off in Konsole](#)
- [Arch Linux logo has a dot in the bottom right corner](#)
- [Incorrect git status in prompt](#)

[oh-my-zsh] theme 'powerlevel10k/powerlevel10k' not found

When opening a terminal, or starting zsh manually, you may encounter this error message:

```
[oh-my-zsh] theme 'powerlevel10k/powerlevel10k' not found
```

1. First, run `typeset -p P9K_VERSION` to check whether Powerlevel10k has been loaded.

- If `typeset -p P9K_VERSION` succeeds and prints something like `typeset P9K_VERSION=1.19.14` (the version could be different), remove the following line from `~/.zshrc`:

```
ZSH_THEME="powerlevel10k/powerlevel10k"
```

- If `typeset -p P9K_VERSION` fails with the error `typeset: no such variable: P9K_VERSION`, run the following command:

```
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ${ZSH_CUSTOM:-$HOME/.oh-my-zsh/custom}/theme
```



2. Restart Zsh with `exec zsh`.

Question mark in prompt

If it looks like a regular `?`, that's normal. It means you have untracked files in the current Git repository. Type `git status` to see the files. You can change this symbol or disable the display of untracked files altogether. Search for `untracked files` in `~/.p10k.zsh`.

FAQ: [What do different symbols in Git status mean?](#)

You can also get a weird-looking question mark in your prompt if your terminal's font is missing some glyphs. See [icons, glyphs or powerline symbols don't render](#).

Icons, glyphs or powerline symbols don't render

Restart your terminal, [install the recommended font](#) and run `p10k configure`.

Sub-pixel imperfections around powerline symbols



There are three imperfections on the screenshot. From left to right:

1. A thin blue line (a sub-pixel gap) between the content of a prompt segment and the following powerline connection.
2. Incorrect alignment of a powerline connection and the following prompt segment. The connection appears shifted to the right.
3. A thin red line below a powerline connection. The connection appears shifted up.

Zsh themes don't have down-to-pixel control over the terminal content. Everything you see on the screen is made of monospace characters. A white powerline prompt segment is made of text on white background followed by U+E0B0 (a right-pointing triangle).

```
print -P 'White background: %K{white}    %k'
print -P 'White triangle:      %F{white}\uE0B0%f'
print -P 'Prompt segment:     %K{white}    %k%F{white}\uE0B0%f'

White background: 
White triangle: 
Prompt segment: 
```

If Powerlevel10k prompt has imperfections around powerline symbols, you'll see exactly the same imperfections with all powerline themes (Agnoster, Powerlevel9k, Powerline, etc.)

There are several things you can try to deal with these imperfections:

- Try [the recommended font](#). If you are already using it, switching to another font may help but is unlikely.
- Change terminal font size one point up or down. For example, in iTerm2 powerline prompt looks perfect at font sizes 11 and 13 but breaks down at 12.
- Enable builtin powerline glyphs in terminal settings if your terminal supports it (iTerm2 does).
- Change font hinting and/or anti-aliasing mode in the terminal settings.
- Shift all text one pixel up/down/left/right if your terminal has an option to do so.
- Try a different terminal.



A more radical solution is to switch to prompt style without background. Type `p10k configure` and select `Lean`. This style has a modern lightweight look. As a bonus, it doesn't suffer from rendering imperfections that afflict powerline-style prompt.

Error: character not in range

Type `echo '\u276F'`. If you get an error saying "zsh: character not in range", your locale doesn't support UTF-8. You need to fix it. If you are running Zsh over SSH, see [this](#). If you are running Zsh locally, Google "set UTF-8 locale in your OS".

Cursor is in the wrong place

Type `echo '\u276F'`. If you get an error saying "zsh: character not in range", see the [previous section](#).

If the `echo` command prints `>` but the cursor is still in the wrong place, install [the recommended font](#) and run `p10k configure`.

If this doesn't help, add `unset ZLE_RPROMPT_INDENT` at the bottom of `~/.zshrc`.

Still having issues? Run the following command to diagnose the problem:

```
(() {
    emulate -L zsh
    setopt err_return no_unset
    local text
    print -rl -- 'Select a part of your prompt from the terminal window and paste it below.' ''
    read -r '?Prompt: ' text
    local -i len=${#text}
    local frame="+${(pl.$len...)}:-+"
    print -lr -- $frame "| $text |" $frame
})
```



If the prompt line aligns with the frame

```
+-----+
| romka@adam ✓ ~/powerlevel10k |
+-----+
```



If the output of the command is aligned for every part of your prompt (left and right), this indicates a bug in the theme or your config. Use this command to diagnose it:

```
print -rl -- ${(eq+)PROMPT} ${(eq+)RPROMPT}
```



Look for `%{...%}` and backslash escapes in the output. If there are any, they are the likely culprits. Open an issue if you get stuck.

If the prompt line is longer than the frame

```
+-----+
| romka@adam ✓ ~/powerlevel10k |
+-----+
```



This is usually caused by a terminal bug or misconfiguration that makes it print ambiguous-width characters as double-width instead of single width. For example, [this issue](#).

If the prompt line is shorter than the frame and is mangled

```
+-----+
| romka@adam ✓ ~/powerlevel10k |
+-----+
```



Note that this prompt is different from the original as it's missing a space after the check mark.

This can be caused by a low-level bug in macOS. See [this issue](#).



This can also happen if prompt contains glyphs designated as "wide" in the Unicode standard and your terminal incorrectly displays them as non-wide. Terminals suffering from this limitation include Konsole, Hyper and the integrated VSCode Terminal. The solution is to use a different terminal or remove all wide glyphs from prompt.

If the prompt line is shorter than the frame and is not mangled

```
+-----+
```



```
| romka@adam ✘ ~/powerlevel10k |  
+-----+
```

This can be caused by misconfigured locale. See [this issue](#).

Prompt wrapping around in a weird way

See [cursor is in the wrong place](#).

Right prompt is in the wrong place

See [cursor is in the wrong place](#).

Configuration wizard runs automatically every time Zsh is started

When Powerlevel10k starts, it automatically runs `p10k configure` if no `POWERLEVEL9K_*` parameters are defined. Based on your prompt style choices, the configuration wizard creates `~/.p10k.zsh` with a bunch of `POWERLEVEL9K_*` parameters in it and adds a line to `~/.zshrc` to source this file. The next time you start Zsh, the configuration wizard shouldn't run automatically. If it does, this means the evaluation of `~/.zshrc` terminates prematurely before it reaches the line that sources `~/.p10k.zsh`. This most often happens due to syntax errors in `~/.zshrc`. These errors get hidden by the configuration wizard screen, so you don't notice them. When you exit configuration wizard, look for error messages. You can also use `POWERLEVEL9K_DISABLE_CONFIGURATION_WIZARD=true` zsh to start Zsh without automatically running the configuration wizard. Once you can see the errors, fix `~/.zshrc` to get rid of them.

Some prompt styles are missing from the configuration wizard

If Zsh version is below 5.7.1 or `COLORTERM` environment variable is neither `24bit` nor `truecolor`, configuration wizard won't offer Pure style with Snazzy color scheme. *Fix:* Install Zsh $\geq 5.7.1$ and use a terminal with truecolor support. Verify with `print -P '%F{#ff0000}red%f'`.

If the terminal can display fewer than 256 colors, configuration wizard preselects Lean style with 8 colors. All other styles require at least 256 colors. *Fix:* Use a terminal with 256 color support and make sure that `TERM` environment variable is set correctly. Verify with `print $terminfo[colors]`.

If there is no UTF-8 locale on the system, configuration wizard won't offer prompt styles that use Unicode characters. *Fix:* Install a UTF-8 locale. Verify with `locale -a`.

Another case in which configuration wizard may not offer Unicode prompt styles is when the `MULTIBYTE` shell option is disabled. *Fix:* Enable the `MULTIBYTE` option, or rather don't disable it (this option is enabled in Zsh by default). Verify with `print -r -- ${options[MULTIBYTE]}`.

When `MULTIBYTE` is enabled and a UTF-8 locale is available, the first few questions asked by the configuration wizard assess capabilities of the terminal font. If your answers indicate that some glyphs don't render correctly, configuration wizard won't offer prompt styles that use them. *Fix:* Restart your terminal and install [the recommended font](#). Verify by running `p10k configure` and checking that all glyphs render correctly.

Cannot install the recommended font

Once you download [the recommended font](#), you can install it just like any other font. Google "how to install fonts on your OS".

Extra or missing spaces in prompt compared to Powerlevel9k

tl;dr: Add `ZLE_RPROMPT_INDENT=0` and `POWERLEVEL9K_LEGACY_ICON_SPACING=true` to `~/.zshrc` to get the same prompt spacing as in Powerlevel9k.

When using Powerlevel10k with a Powerlevel9k config, you might get additional spaces in prompt here and there. These come in two flavors.

Extra space without background on the right side of right prompt

tl;dr: Add `ZLE_RPROMPT_INDENT=0` to `~/.zshrc` to get rid of that space.

From [Zsh documentation](#):

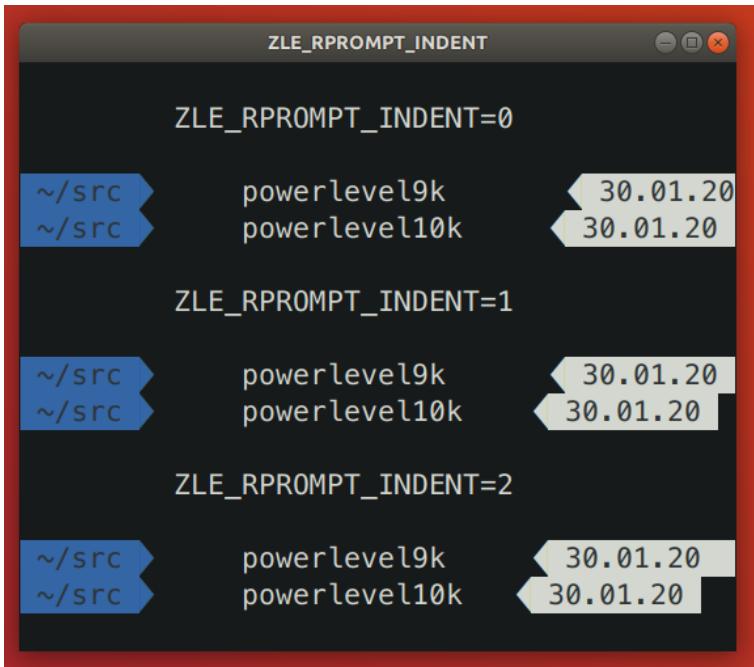
```
ZLE_RPROMPT_INDENT <s>
```



If set, used to give the indentation between the right hand side of the right prompt in the line editor as given by `RPS1` or `RPROMPT` and the right hand side of the screen. If not set, the value `1` is used.

Typically this will be used to set the value to `0` so that the prompt appears flush with the right hand side of the screen.

Powerlevel10k respects this parameter. If you set `ZLE_RPROMPT_INDENT=1` (or leave it unset, which is the same thing as setting it to `1`) you'll get an empty space to the right of right prompt. If you set `ZLE_RPROMPT_INDENT=0`, your prompt will go to the edge of the terminal. This is how it works in every theme except Powerlevel9k.



Powerlevel9k issue: [powerlevel9k#1292](#). It's been fixed in the development branch of Powerlevel9k but the fix hasn't yet made it to master .

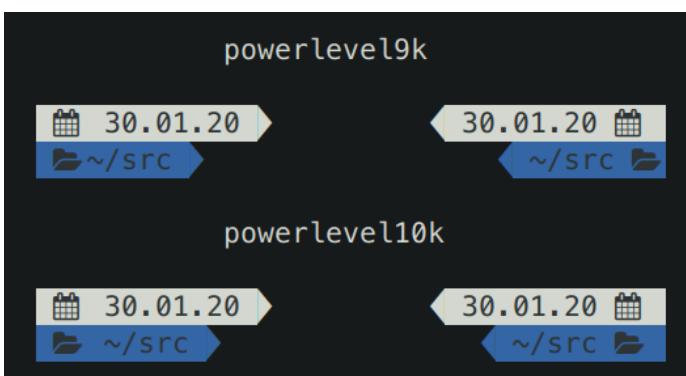
Add `ZLE_RPROMPT_INDENT=0` to `~/.zshrc` to get the same spacing on the right edge of prompt as in Powerlevel9k.

Note: Several versions of Zsh have bugs that get triggered when you set `ZLE_RPROMPT_INDENT=0`. Powerlevel10k can work around these bugs when using powerline prompt style. If you notice visual artifacts in prompt, or wrong cursor position, try removing `ZLE_RPROMPT_INDENT` from `~/.zshrc`.

Extra or missing spaces around icons

tl;dr: Add `POWERLEVEL9K_LEGACY_ICON_SPACING=true` to `~/.zshrc` to get the same spacing around icons as in Powerlevel9k.

Spacing around icons in Powerlevel9k is inconsistent.



This inconsistency is a constant source of annoyance, so it was fixed in Powerlevel10k. You can add `POWERLEVEL9K_LEGACY_ICON_SPACING=true` to `~/.zshrc` to get the same spacing around icons as in Powerlevel9k.

Note: It's not a good idea to define `POWERLEVEL9K_LEGACY_ICON_SPACING` when using `p10k configure` .

Weird things happen after typing `source ~/.zshrc`

It's almost always a bad idea to run `source ~/.zshrc`, whether you are using Powerlevel10k or not. This command may result in random errors, misbehaving code and progressive slowdown of Zsh.

If you've made changes to `~/.zshrc` or to files sourced by it, restart Zsh to apply them. The most reliable way to do this is to type `exit` and then start a new Zsh session. You can also use `exec zsh`. While not exactly equivalent to complete Zsh restart, this command is much more reliable than `source ~/.zshrc`.

Transient prompt stops working after some time

See [weird things happen after typing `source ~/.zshrc`](#).

Cannot make Powerlevel10k work with my plugin manager

If the [installation instructions](#) didn't work for you, try disabling your current theme (so that you end up with no theme) and then installing Powerlevel10k manually.

1. Disable the current theme in your framework / plugin manager.

- **oh-my-zsh:** Open `~/.zshrc` and remove the line that sets `ZSH_THEME`. It might look like this:
`ZSH_THEME="powerlevel9k/powerlevel9k".`
- **zplug:** Open `~/.zshrc` and remove the `zplug` command that refers to your current theme. For example, if you are currently using Powerlevel9k, look for `zplug bhilburn/powerlevel9k, use:powerlevel9k.zsh-theme`.
- **prezto:** Open `~/.zpreztorc` and put `zstyle :prezto:module:prompt theme off` in it. Remove any other command that sets `theme` such as `zstyle :prezto:module:prompt theme powerlevel9k`.
- **antigen:** Open `~/.zshrc` and remove the line that sets `antigen theme`. It might look like this: `antigen theme powerlevel9k/powerlevel9k`.

2. Install Powerlevel10k manually.

```
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ~/powerlevel10k
echo 'source ~/powerlevel10k/powerlevel10k.zsh-theme' >> ~/.zshrc
```



This method of installation won't make anything slower or otherwise sub-par.

Directory is difficult to see in prompt when using Rainbow style

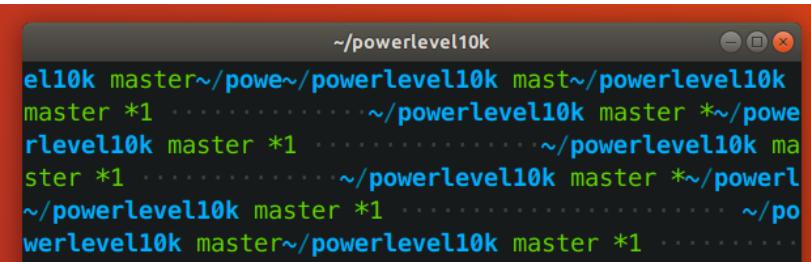
In Rainbow style the current working directory is shown with bright white text on blue background. The white is fixed and always look the same but the appearance of "blue" is defined by your terminal color palette. If it's very light, it may be difficult to see white text on it.

There are several ways to fix this.

- Type `p10k configure` and choose a more readable prompt style.
- [Change terminal color palette](#). Try Tango Dark or Solarized Dark, or change just the "blue" color.
- [Change directory background and/or foreground color](#). The parameters you are looking for are called `POWERLEVEL9K_DIR_BACKGROUND`, `POWERLEVEL9K_DIR_FOREGROUND`, `POWERLEVEL9K_DIR_SHORTENED_FOREGROUND`, `POWERLEVEL9K_DIR_ANCHOR_FOREGROUND` and `POWERLEVEL9K_DIR_ANCHOR_BOLD`. You can find them in `~/.p10k.zsh`.

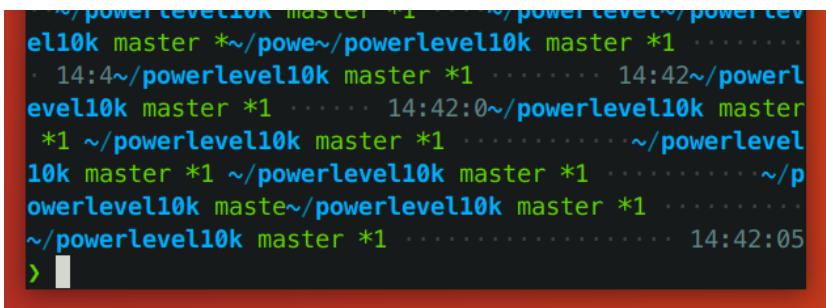
Horrific mess when resizing terminal window

When you resize a terminal window horizontally back and forth a few times, you might see this ugly picture.



The screenshot shows a terminal window with a red border. Inside, there are multiple overlapping instances of the directory prompt 'el10k master~/powerlevel10k'. Each instance has a different background color (blue, green, yellow, etc.) and foreground color (white, black, etc.), creating a chaotic visual effect where the text is illegible due to color bleed-through.





```
~/powerlevel10k master *1 ~~/powerlev~ powerlevell0k master *1  
· 14:4~~/powerlevel10k master *1 ..... 14:42~/powerl  
evel10k master *1 ..... 14:42:0~/powerlevel10k master  
*1 ~/powerlevel10k master *1 ..... ~/p  
owerlevel10k master ~~/powerlevel10k master *1 .....  
~/powerlevel10k master *1 ..... 14:42:05  
> |
```

tl;dr: This issue arises when a terminal reflows Zsh prompt upon resizing. It isn't specific to Powerlevel10k. See [mitigation](#).

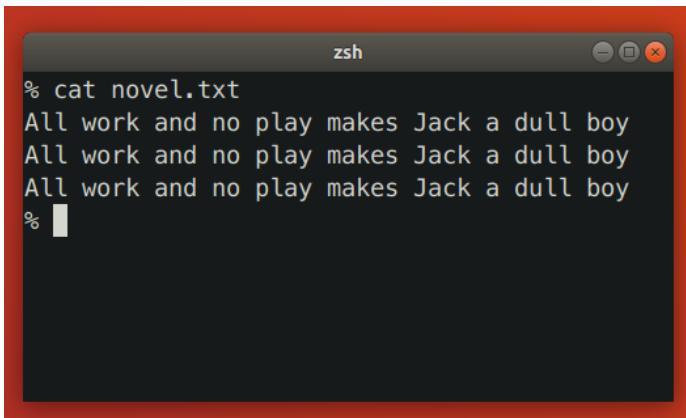
Note: This section [used to say](#) that the problem is caused by a bug in Zsh. While it's true that it's possible to avoid the problem in many circumstances by modifying Zsh, it cannot be completely resolved this way. Thus it's unfair to pin the blame on Zsh.

The anatomy of the problem

The issue is manifested when the vertical distance between the start of the current prompt and the cursor (henceforth `vd`) changes when the terminal window is resized.

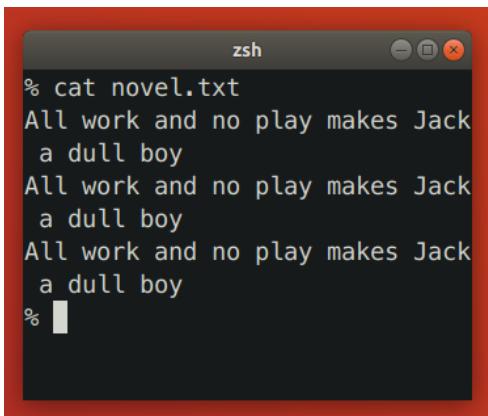
When a terminal window gets shrunk horizontally, there are two ways for a terminal to handle long lines that no longer fit: *reflow* or *truncate*.

Terminal content before shrinking:



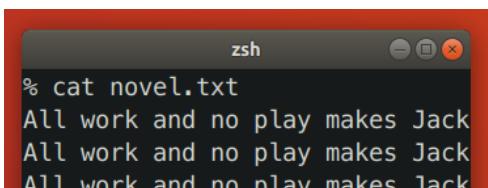
```
% cat novel.txt  
All work and no play makes Jack a dull boy  
All work and no play makes Jack a dull boy  
All work and no play makes Jack a dull boy  
%
```

Terminal reflows text when shrinking:



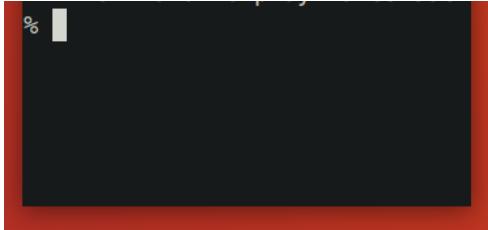
```
% cat novel.txt  
All work and no play makes Jack  
a dull boy  
All work and no play makes Jack  
a dull boy  
All work and no play makes Jack  
a dull boy  
%
```

Terminal truncates text when shrinking:



```
% cat novel.txt  
All work and no play makes Jack  
All work and no play makes Jack  
All work and no play makes Jack
```





Reflowing strategy can change the height of terminal content. If such content happens to be between the start of the current prompt and the cursor, Zsh will print prompt on the wrong line. Truncation strategy never changes the height of terminal content, so it doesn't trigger this issue.

Let's see how the issue plays out in slow motion. We'll start by launching `zsh -f` and pasting the following code:

```
function pause() { read -s }
functions -M pause 0

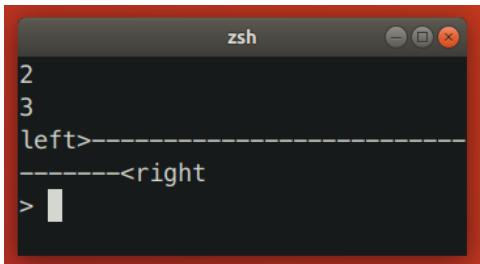
reset
print -l {1..3}
setopt prompt_subst
PROMPT=$'${${((pause()))}+}left>${(pl.$((COLUMNS-12))...)}<right\n> '
```



When `PROMPT` gets expanded, it calls `pause` to let us observe the state of the terminal. Here's the initial state:



Zsh keeps track of the cursor position relative to the start of the current prompt. In this case it knows that the cursor is one line below the start of the prompt. When we shrink the terminal window, it looks like this:



At this point the terminal sends `SIGWINCH` to Zsh to notify it about changes in the terminal dimensions. Note that this signal is sent *after* the content of the terminal has been refloated.

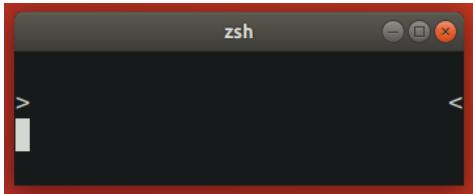
When Zsh receives `SIGWINCH`, it attempts to erase the current prompt and print it anew. It goes to the position where it *thinks* the current prompt is -- one line above the cursor (!) -- erases all terminal content that follows and prints reexpanded prompt there. However, after resizing prompt is no longer one line above the cursor. It's two lines above! Zsh ends up printing new prompt one line too low.



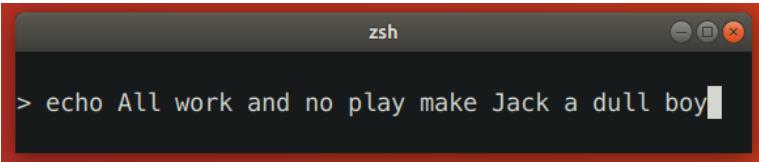
In this case we ended up with unwanted junk content because `vp` has *increased*. When you make terminal window wider, `vp` can also *decrease*, which would result in the new prompt being printed higher than intended, potentially erasing useful content in the process.

Here are a few more examples where shrinking terminal window increased `vp`.

- Simple one-line left prompt with right prompt. No `prompt_subst`. Note that the cursor is below the prompt line (hit `ESC-ENTER` to get it there).



- Simple one-line left prompt. No `prompt_subst`, no right prompt. Here `vp` is bound to increase upon terminal shrinking due to the command line wrapping around.



Zsh patch

[This Zsh patch](#) fixes the issue on some terminals. The idea behind the patch is to use `sc` (save cursor) terminal capability before printing prompt and `rc` (restore cursor) to move cursor back to the original position when prompt needs to be refreshed.

The patch works only on terminals that reflow saved cursor position together with text when the terminal window is resized. The patch has no observable effect on terminals that don't reflow text on resize (both patched and unpatched Zsh behave correctly) and on terminals that reflow text but not the saved cursor position (both patched and unpatched Zsh redraw prompt at the same incorrect position). In other words, the patch fixes the resizing issue on some terminals while keeping the behavior unchanged on others.

There are two alternative approaches to patching Zsh that may seem to work at first glance but in fact don't:

- Instead of `sc`, use `u7` terminal capability to query the current cursor position and then `cup` to go back to it. This doesn't work because the absolute position of the start of the current prompt changes when text gets reflowed.
- Recompute `vp` based on new terminal dimensions before attempting to refresh prompt. This doesn't work because Zsh doesn't know whether terminal reflows text or truncates it. If Zsh could somehow know that the terminal reflows text, this approach still wouldn't work on terminals that continuously reflow text and rapid-fire `SIGWINCH` when the window is being resized. In such environment real terminal dimensions go out of sync with what Zsh thinks the dimensions are.

There is no ETA for the patch making its way into upstream Zsh. See [discussion](#).

Mitigation

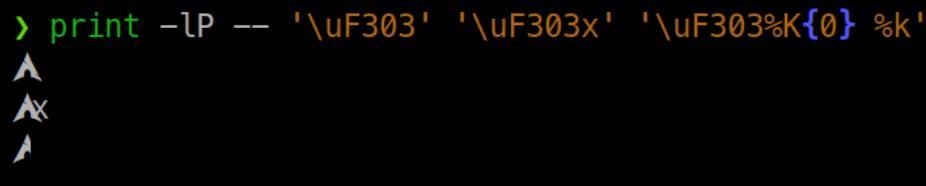
There are a few mitigation options for this issue.

- Use [kitty](#) terminal version $\geq 0.24.0$ and enable terminal-shell integration in Powerlevel10k by defining `POWERLEVEL9K_TERM_SHELL_INTEGRATION=true` in `~/.p10k.zsh`.
- Apply [the patch](#) and [rebuild Zsh from source](#). It won't help if you are using Alacritty, kitty or some other terminal that reflows text on resize but doesn't reflow saved cursor position. On such terminals the patch will have no visible effect.
- Disable text reflowing on window resize in terminal settings. If your terminal doesn't have this setting, try a different terminal.
- Avoid long lines between the start of prompt and cursor.
 - Disable ruler with `POWERLEVEL9K_SHOW_RULER=false`.
 - Disable prompt connection with `POWERLEVEL9K_MULTILINE_FIRST_PROMPT_GAP_CHAR=' '`.
 - Disable right frame with `POWERLEVEL9K_MULTILINE_FIRST_PROMPT_SUFFIX=''`, `POWERLEVEL9K_MULTILINE_NEWLINE_PROMPT_SUFFIX=''` and `POWERLEVEL9K_MULTILINE_LAST_PROMPT_SUFFIX=''`.
- iv. Set `POWERLEVEL9K_RIGHT_PROMPT_ELEMENTS=()`. Right prompt on the last prompt line will cause resizing issues only when the cursor is below it. This isn't very common, so you might want to keep some elements in `POWERLEVEL9K_RIGHT_PROMPT_ELEMENTS` provided that none of them are succeeded by `newline`.

Icons cut off in Konsole

When using Konsole with a non-monospace font, icons may be cut off on the right side. Here "non-monospace" refers to any font with glyphs wider than a single column, or wider than two columns for glyphs designated as "wide" in the Unicode standard.





```
> print -lP -- '\uF303' '\uF303x' '\uF303%K{0} %k'  
▲  
▲X  
▲
```

The last line on the screenshot shows a cut off Arch Linux logo.

There are several mitigation options for this issue.

1. Use a different terminal. Konsole is the only terminal that exhibits this behavior.
2. Use a monospace font.
3. Manually add an extra space after the icon that gets cut off. For example, if the content of `os_icon` prompt segment gets cut off open `~/.p10k.zsh`, search for `POWERLEVEL9K_OS_ICON_CONTENT_EXPANSION` and change it as follows:

```
typeset -g POWERLEVEL9K_OS_ICON_CONTENT_EXPANSION='${P9K_CONTENT} ' # extra space at the end
```

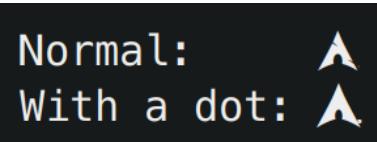
4. Use a different icon that is monospace. For example, if Arch Linux logo gets cut off, add the following parameter to `~/.p10k.zsh`

```
typeset -g POWERLEVEL9K_LINUX_ARCH_ICON='Arch' # plain "Arch" in place of a logo
```

5. Disable the display of the icon that gets cut off. For example, if the content of `os_icon` prompt segment gets cut off, open `~/.p10k.zsh` and remove `os_icon` from `POWERLEVEL9K_LEFT_PROMPT_ELEMENTS` and `POWERLEVEL9K_RIGHT_PROMPT_ELEMENTS`.

Note: [Non-monospace fonts are not officially supported by Konsole](#).

Arch Linux logo has a dot in the bottom right corner



Some fonts have this incorrect dotted icon in bold typeface. There are two ways to fix this issue.

1. Use a font with a correct Arch Linux logo in bold typeface. For example, [the recommended Powerlevel10k font](#).
2. Display the icon in regular (non-bold) typeface. To do this, open `~/.p10k.zsh`, search for `POWERLEVEL9K_OS_ICON_CONTENT_EXPANSION` and remove `%B` from its value.

```
typeset -g POWERLEVEL9K_OS_ICON_CONTENT_EXPANSION='${P9K_CONTENT}' # not bold
```

Incorrect git status in prompt

Powerlevel10k uses [gitstatusd](#) to inspect the state of git repositories. The project relies on the [libgit2](#) library, which has some gaps in its implementation. Under some conditions, this may result in discrepancies between the real state of a git repository (reflected by `git status`) and what gets shown in the Powerlevel10k prompt.

Most notably, [libgit2 does not support skipHash](#). If you see incorrect git status in prompt, run `git config -l` and check whether `skipHash` is enabled. If it is, consider disabling it. Keep in mind that `skipHash` may be implicitly enabled when activating certain git features, such as `manyFiles`.



Releases 33

 v1.20.0 Latest
2 weeks ago

[+ 32 releases](#)

Contributors 259

[+ 245 contributors](#)

Languages

● Shell 85.7% ● C++ 14.1% ● Makefile 0.2%



Style Guide

Style Guide

This guide documents how to use the official GJS ESLint configuration, as well as other preferred styles that can't be expressed by a linter configuration.

It also includes a basic introduction to setting up a project to use `.eslintrc.yml` and `.editorconfig` files, to help reduce manual work for developers.

ESLint

GNOME Shell includes one additional global variable called `global`.

ESLint is a well known linter and static analysis tool for JavaScript, used by both GJS and GNOME Shell. It's used by many projects to maintain code quality, enforce coding standards, catch potential errors, and improve code consistency.

The recommended configuration includes rules for static analysis, deprecated syntax and a list of all the global variables for the environment. Put the `.eslintrc.yml` in the root of your project directory, and your IDE will provide real-time diagnostics and warnings.

```
::: details .eslintrc.yml @code{4-109} yml :::
```

ESLint is transitioning to a new flat configuration that uses ES Modules. To use this configuration, be sure your project has a `package.json` file with `"sourceType": "module"`.

```
::: details eslint.config.js @code{4-143} js :::
```

Continuous Integration

In most projects, it is recommended practice to run tests on every pull request before merging into the main branch. Below are two example CI configurations for running ESLint with GitLab and GitHub.

```
::: details GitLab (.gitlab-ci.yml) @code yml :::
```

```
::: details GitHub (.github/workflows/eslint.yml) @code yml :::
```

Prettier

Prettier is another popular tool for JavaScript projects, renowned for its lack of options. It focuses specifically on formatting code, and won't catch logic errors or anti-patterns like ESLint.

Below is a sample configuration (with almost all available options), set to resemble the code style used by many GJS applications:

```
@code yml
```

EditorConfig

EditorConfig is a more general formatting tool, targeted directly at IDEs like GNOME Builder and VSCode. It's used to tell an editor to trim trailing whitespace, what indentation to use, and other similar preferences.

Below is the `.editorconfig` file used in the GJS project:

```
@code{6-17} ini
```

Code Conventions

The following guidelines are general recommendations and coding conventions followed by many GJS projects. As general rule, you should take advantage of modern language features, both in JavaScript and GJS.

Files and Imports

GJS has supported ESModules since GNOME 40, and GNOME Shell extensions are required to use them since GNOME 45.

JavaScript file names should be `lowerCamelCase` with a `.js` extension, while directories should be short and `lowercase`:

```
js/misc/extensionSystem.js  
js/ui/panel.js
```

Use `PascalCase` when importing modules and classes

```
import * as Util from 'resource:///gjs/guide/Example/js/util.js';
```

Keep library, module and local imports separated by a single line.

```
import Gio from 'gi://Gio';  
  
import * as Main from 'resource:///org/gnome/shell/ui/main.js';  
  
import * as Util from './lib/util.js';
```

GObject

See GObject Basics for more details about using GObject in JavaScript.

Properties When possible, set all properties when constructing an object, which is cleaner and avoids extra property notifications.

@code{7-9} js:no-line-numbers

Using `camelCase` property accessors is preferred by many GNOME projects. GJS can automatically convert GObject property names, except when used as a string.

@code{11-14} js:no-line-numbers

Asynchronous Operations Use `Gio._promisify()` to enable `async/await` with asynchronous methods in platform libraries:

```
import GLib from 'gi://GLib';
import Gio from 'gi://Gio';

Gio._promisify(Gio.File.prototype, 'delete_async');

const file = Gio.File.new_for_path('file.txt');
await file.delete_async(GLib.PRIORITY_DEFAULT, null /* cancellable */);
```

JavaScript

Variables and Exports Use `const` when the value will be bound to a static value, and `let` when you need a mutable variable:

@code{10-17} js:no-line-numbers

The `var` statement should be avoided, since it has unexpected behavior like hoisting. Although it was used in older code to make members of a script public, `export` should now be used in all new code:

@code{20-26} js:no-line-numbers

Classes and Functions Define classes with `class` and override the standard `constructor()` when subclassing GObject classes:

@code{32-46} js:no-line-numbers

Use arrow functions for inline callbacks and `Function.prototype.bind()` for larger functions.

@code{49-66} js:no-line-numbers

Name

[systemd.index](#) — List all manpages from the systemd project

3

[30-systemd-environment-d-generator\(8\)](#) — Load variables specified by environment.d

B

[binfmt.d\(5\)](#) — Configure additional binary formats for executables at boot
[bootctl\(1\)](#) — Control EFI firmware boot settings and manage boot loader
[bootup\(7\)](#) — System bootup process
[busctl\(1\)](#) — Introspect the bus

C

[coredump.conf\(5\)](#) — Core dump storage configuration files
[coredump.conf.d\(5\)](#) — Core dump storage configuration files
[coredumpctl\(1\)](#) — Retrieve and process saved core dumps and metadata
[crypttab\(5\)](#) — Configuration for encrypted block devices

D

[daemon\(7\)](#) — Writing and packaging system daemons
[dnssec-trust-anchors.d\(5\)](#) — DNSSEC trust anchor configuration files

E

[environment.d\(5\)](#) — Definition of user service environment
[extension-release\(5\)](#) — Operating system identification

F

[file-hierarchy\(7\)](#) — File system hierarchy overview



H

[halt\(8\)](#) — Power off, reboot, or halt the machine
[homectl\(1\)](#) — Create, remove, change or inspect home directories
[homed.conf\(5\)](#) — Home area/user account manager configuration files
[homed.conf.d\(5\)](#) — Home area/user account manager configuration files

[hostname\(5\)](#) — Local hostname configuration file

[hostnamectl\(1\)](#) — Control the system hostname

[hwdb\(7\)](#) — Hardware Database

I

[init\(1\)](#) — systemd system and service manager

[initrd-release\(5\)](#) — Operating system identification

[integritytab\(5\)](#) — Configuration for integrity block devices

[iocost.conf\(5\)](#) — Configuration files for the iocost solution manager

J

[journal-remote.conf\(5\)](#) — Configuration files for the service accepting remote journal uploads

[journal-remote.conf.d\(5\)](#) — Configuration files for the service accepting remote journal uploads

[journal-upload.conf\(5\)](#) — Configuration files for the journal upload service

[journal-upload.conf.d\(5\)](#) — Configuration files for the journal upload service

[journalctl\(1\)](#) — Print log entries from the systemd journal

[journald.conf\(5\)](#) — Journal service configuration files

[journald.conf.d\(5\)](#) — Journal service configuration files

[journald@.conf\(5\)](#) — Journal service configuration files

K

[kernel-command-line\(7\)](#) — Kernel command line parameters

[kernel-install\(8\)](#) — Add and remove kernel and initrd images to and from /boot

L

[libnss_myhostname.so.2\(8\)](#) — Hostname resolution for the locally configured system hostname

[libnss_mymachines.so.2\(8\)](#) — Hostname resolution for local container instances

[libnss_resolve.so.2\(8\)](#) — Hostname resolution via `systemd-resolved.service`

[libnss_systemd.so.2\(8\)](#) — UNIX user and group name resolution for user/group lookup via Varlink

[libsystemd\(3\)](#) — Functions for implementing services and interacting with systemd

[libudev\(3\)](#) — API for enumerating and introspecting local devices

[linuxaa64.efi.stub\(7\)](#) — A simple UEFI kernel boot stub

[linuxia32.efi.stub\(7\)](#) — A simple UEFI kernel boot stub

[linuxx64.efi.stub\(7\)](#) — A simple UEFI kernel boot stub

[loader.conf\(5\)](#) — Configuration file for `systemd-boot`

[locale.conf\(5\)](#) — Configuration file for locale settings

[localectl\(1\)](#) — Control the system locale and keyboard layout settings

[localtime\(5\)](#) — Local timezone configuration file

[loginctl\(1\)](#) — Control the systemd login manager

[logind.conf\(5\)](#) — Login manager configuration files
[logind.conf.d\(5\)](#) — Login manager configuration files

M

[machine-id\(5\)](#) — Local machine ID configuration file
[machine-info\(5\)](#) — Local machine information file
[machinectl\(1\)](#) — Control the systemd machine manager
[modules-load.d\(5\)](#) — Configure kernel modules to load at boot
[mount.ddi\(1\)](#) — Dissect Discoverable Disk Images (DDIs)

N

[networkctl\(1\)](#) — Query or modify the status of network links
[networkd.conf\(5\)](#) — Global Network configuration files
[networkd.conf.d\(5\)](#) — Global Network configuration files
[nss-myhostname\(8\)](#) — Hostname resolution for the locally configured system hostname
[nss-mymachines\(8\)](#) — Hostname resolution for local container instances
[nss-resolve\(8\)](#) — Hostname resolution via `systemd-resolved.service`
[nss-systemd\(8\)](#) — UNIX user and group name resolution for user/group lookup via Varlink

O

[oomctl\(1\)](#) — Analyze the state stored in `systemd-oomd`
[oomd.conf\(5\)](#) — Global `systemd-oomd` configuration files
[oomd.conf.d\(5\)](#) — Global `systemd-oomd` configuration files
[org.freedesktop.home1\(5\)](#) — The D-Bus interface of `systemd-homed`
[org.freedesktop.hostname1\(5\)](#) — The D-Bus interface of `systemd-hostnamed`
[org.freedesktop.import1\(5\)](#) — The D-Bus interface of `systemd-importd`
[org.freedesktop.locale1\(5\)](#) — The D-Bus interface of `systemd-located`
[org.freedesktop.LogControl1\(5\)](#) — D-Bus interface to query and set logging configuration
[org.freedesktop.login1\(5\)](#) — The D-Bus interface of `systemd-logind`
[org.freedesktop.machine1\(5\)](#) — The D-Bus interface of `systemd-machined`
[org.freedesktop.network1\(5\)](#) — The D-Bus interface of `systemd-networkd`
[org.freedesktop.oom1\(5\)](#) — The D-Bus interface of `systemd-oomd`
[org.freedesktop.portable1\(5\)](#) — The D-Bus interface of `systemd-portabled`
[org.freedesktop.resolve1\(5\)](#) — The D-Bus interface of `systemd-resolved`
[org.freedesktop.systemd1\(5\)](#) — The D-Bus interface of `systemd`
[org.freedesktop.timedate1\(5\)](#) — The D-Bus interface of `systemd-timedated`
[os-release\(5\)](#) — Operating system identification



P

[pam_systemd\(8\)](#) — Register user sessions in the `systemd` login manager

[**pam_systemd_home\(8\)**](#) — Authenticate users and mount home directories via systemd-homed.service
[**pam_systemd_loadkey\(8\)**](#) — Read password from kernel keyring and set it as PAM authtok
[**portablectl\(1\)**](#) — Attach, detach or inspect portable service images
[**poweroff\(8\)**](#) — Power off, reboot, or halt the machine
[**pstore.conf\(5\)**](#) — PStore configuration file
[**pstore.conf.d\(5\)**](#) — PStore configuration file

R

[**rc-local.service\(8\)**](#) — Compatibility generator and service to start /etc/rc.local during boot
[**reboot\(8\)**](#) — Power off, reboot, or halt the machine
[**repard\(5\)**](#) — Partition Definition Files for Automatic Boot-Time Repartitioning
[**resolvconf\(1\)**](#) — Resolve domain names, IPV4 and IPv6 addresses, DNS resource records, and services; introspect and reconfigure the DNS resolver
[**resolvect\(1\)**](#) — Resolve domain names, IPV4 and IPv6 addresses, DNS resource records, and services; introspect and reconfigure the DNS resolver
[**resolved.conf\(5\)**](#) — Network Name Resolution configuration files
[**resolved.conf.d\(5\)**](#) — Network Name Resolution configuration files
[**runlevel\(8\)**](#) — Print previous and current SysV runlevel

S

[**sd-boot\(7\)**](#) — A simple UEFI boot manager
[**sd-bus\(3\)**](#) — A lightweight D-Bus IPC client library
[**sd-bus-errors\(3\)**](#) — Standard D-Bus error names
[**sd-daemon\(3\)**](#) — APIs for new-style daemons
[**sd-device\(3\)**](#) — API for enumerating and introspecting local devices
[**sd-event\(3\)**](#) — A generic event loop implementation
[**sd-hwdb\(3\)**](#) — Read-only access to the hardware description database
[**sd-id128\(3\)**](#) — APIs for processing 128-bit IDs
[**sd-journal\(3\)**](#) — APIs for submitting and querying log entries to and from the journal
[**sd-login\(3\)**](#) — APIs for tracking logins
[**sd-stub\(7\)**](#) — A simple UEFI kernel boot stub
[**SD_ALERT\(3\)**](#) — APIs for new-style daemons
[**sd_booted\(3\)**](#) — Test whether the system is running the systemd init system
[**sd_bus_add_fallback\(3\)**](#) — Declare properties and methods for a D-Bus path
[**sd_bus_add_fallback_vtable\(3\)**](#) — Declare properties and methods for a D-Bus path 
[**sd_bus_add_filter\(3\)**](#) — Declare properties and methods for a D-Bus path
[**sd_bus_add_match\(3\)**](#) — Add a match rule for incoming message dispatching
[**sd_bus_add_match_async\(3\)**](#) — Add a match rule for incoming message dispatching
[**sd_bus_add_node_enumerator\(3\)**](#) — Add a node enumerator for a D-Bus object path prefix
[**sd_bus_add_object\(3\)**](#) — Declare properties and methods for a D-Bus path
[**sd_bus_add_object_manager\(3\)**](#) — Add a D-Bus object manager for a D-Bus object subtree

[`sd_bus_add_object_vtable\(3\)`](#) — Declare properties and methods for a D-Bus path
[`sd_bus_attach_event\(3\)`](#) — Attach a bus connection object to an event loop
[`sd_bus_call\(3\)`](#) — Invoke a D-Bus method call
[`sd_bus_call_async\(3\)`](#) — Invoke a D-Bus method call
[`sd_bus_call_method\(3\)`](#) — Initialize a bus message object and invoke the corresponding D-Bus method call
[`sd_bus_call_method_async\(3\)`](#) — Initialize a bus message object and invoke the corresponding D-Bus method call
[`sd_bus_call_methodv\(3\)`](#) — Initialize a bus message object and invoke the corresponding D-Bus method call
[`sd_bus_can_send\(3\)`](#) — Check which types can be sent over a bus object
[`sd_bus_close\(3\)`](#) — Close and flush a bus connection
[`sd_bus_close_unref\(3\)`](#) — Create a new bus object and create or destroy references to it
[`sd_bus_close_unrefp\(3\)`](#) — Create a new bus object and create or destroy references to it
[`sd_buscreds_get_audit_login_uid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_audit_session_id\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_augmented_mask\(3\)`](#) — Retrieve credentials object for the specified PID
[`sd_buscreds_get_cgroup\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_cmdline\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_comm\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_description\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_egid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_euid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_exe\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_fsgid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_fsuid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_gid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_mask\(3\)`](#) — Retrieve credentials object for the specified PID
[`sd_buscreds_get_owner_uid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_pid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_ppid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_selinux_context\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_session\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_sgid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_slice\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_suid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_supplementary_gids\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_tid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_tid_comm\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_tty\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_uid\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_unique_name\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_unit\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_user_slice\(3\)`](#) — Retrieve fields from a credentials object



[`sd_buscreds_get_user_unit\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_get_well_known_names\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_has_bounding_cap\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_has_effective_cap\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_has_inheritable_cap\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_has_permitted_cap\(3\)`](#) — Retrieve fields from a credentials object
[`sd_buscreds_new_from_pid\(3\)`](#) — Retrieve credentials object for the specified PID
[`sd_buscreds_ref\(3\)`](#) — Retrieve credentials object for the specified PID
[`sd_buscreds_unref\(3\)`](#) — Retrieve credentials object for the specified PID
[`sd_buscreds_unrefp\(3\)`](#) — Retrieve credentials object for the specified PID
[`sd_bus_default\(3\)`](#) — Acquire a connection to a system or user bus
[`sd_bus_default_flush_close\(3\)`](#) — Close and flush a bus connection
[`sd_bus_default_system\(3\)`](#) — Acquire a connection to a system or user bus
[`sd_bus_default_user\(3\)`](#) — Acquire a connection to a system or user bus
[`sd_bus_destroy_t\(3\)`](#) — Define the callback function for resource cleanup
[`sd_bus_detach_event\(3\)`](#) — Attach a bus connection object to an event loop
[`sd_bus_emit_interfaces_added\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_interfaces_added_strv\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_interfaces_removed\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_interfaces_removed_strv\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_object_added\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_object_removed\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_properties_changed\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_properties_changed_strv\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_signal\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_signal_to\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_signal_tov\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_emit_signalf\(3\)`](#) — Convenience functions for emitting (standard) D-Bus signals
[`sd_bus_enqueue_for_read\(3\)`](#) — Re-enqueue a bus message on a bus connection, for reading
[`sd_bus_error\(3\)`](#) — sd-bus error handling
[`SD_BUS_ERROR_ACCESS_DENIED\(3\)`](#) — Standard D-Bus error names
[`sd_bus_error_add_map\(3\)`](#) — Additional sd-dbus error mappings
[`SD_BUS_ERROR_ADDRESS_IN_USE\(3\)`](#) — Standard D-Bus error names
[`SD_BUS_ERROR_AUTH_FAILED\(3\)`](#) — Standard D-Bus error names
[`SD_BUS_ERROR_BAD_ADDRESS\(3\)`](#) — Standard D-Bus error names
[`sd_bus_error_copy\(3\)`](#) — sd-bus error handling
[`SD_BUS_ERROR_DISCONNECTED\(3\)`](#) — Standard D-Bus error names



[SD_BUS_ERROR_END\(3\)](#) — Additional sd-dbus error mappings
[SD_BUS_ERROR_FAILED\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_FILE_EXISTS\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_FILE_NOT_FOUND\(3\)](#) — Standard D-Bus error names
[sd_bus_error_free\(3\)](#) — sd-bus error handling
[sd_bus_error_get_errno\(3\)](#) — sd-bus error handling
[sd_bus_error_has_name\(3\)](#) — sd-bus error handling
[sd_bus_error_has_names\(3\)](#) — sd-bus error handling
[sd_bus_error_has_names_sentinel\(3\)](#) — sd-bus error handling
[SD_BUS_ERROR_INCONSISTENT_MESSAGE\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_INTERACTIVE_AUTHORIZATION_REQUIRED\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_INVALID_ARGS\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_INVALID_FILE_CONTENT\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_INVALID_SIGNATURE\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_IO_ERROR\(3\)](#) — Standard D-Bus error names
[sd_bus_error_is_set\(3\)](#) — sd-bus error handling
[SD_BUS_ERROR_LIMITS_EXCEEDED\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_MAKE_CONST\(3\)](#) — sd-bus error handling
[sd_bus_error_map\(3\)](#) — Additional sd-dbus error mappings
[SD_BUS_ERROR_MAP\(3\)](#) — Additional sd-dbus error mappings
[SD_BUS_ERROR_MATCH_RULE_INVALID\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_MATCH_RULE_NOT_FOUND\(3\)](#) — Standard D-Bus error names
[sd_bus_error_move\(3\)](#) — sd-bus error handling
[SD_BUS_ERROR_NAME_HAS_NO_OWNER\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_NO_MEMORY\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_NO_NETWORK\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_NO_REPLY\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_NO_SERVER\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_NOT_SUPPORTED\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_NULL\(3\)](#) — sd-bus error handling
[SD_BUS_ERROR_OBJECT_PATH_IN_USE\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_PROPERTY_READ_ONLY\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_SELINUX_SECURITY_CONTEXT_UNKNOWN\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_SERVICE_UNKNOWN\(3\)](#) — Standard D-Bus error names
[sd_bus_error_set\(3\)](#) — sd-bus error handling
[sd_bus_error_set_const\(3\)](#) — sd-bus error handling
[sd_bus_error_set_errno\(3\)](#) — sd-bus error handling
[sd_bus_error_set_errnof\(3\)](#) — sd-bus error handling
[sd_bus_error_set_errnofv\(3\)](#) — sd-bus error handling
[sd_bus_error_setf\(3\)](#) — sd-bus error handling
[sd_bus_error_setfv\(3\)](#) — sd-bus error handling
[SD_BUS_ERROR_TIMED_OUT\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_TIMEOUT\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_UNIX_PROCESS_ID_UNKNOWN\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_UNKNOWN_INTERFACE\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_UNKNOWN_METHOD\(3\)](#) — Standard D-Bus error names
[SD_BUS_ERROR_UNKNOWN_OBJECT\(3\)](#) — Standard D-Bus error names



[**SD_BUS_ERROR_UNKNOWN_PROPERTY\(3\)**](#) — Standard D-Bus error names
[**sd_bus_flush\(3\)**](#) — Close and flush a bus connection
[**sd_bus_flush_close_unref\(3\)**](#) — Create a new bus object and create or destroy references to it
[**sd_bus_flush_close_unrefp\(3\)**](#) — Create a new bus object and create or destroy references to it
[**sd_bus_get_address\(3\)**](#) — Set or query the address of the bus connection
[**sd_bus_get_allow_interactive_authorization\(3\)**](#) — Set or query properties of a bus object
[**sd_bus_get_bus_id\(3\)**](#) — Configure connection mode for a bus object
[**sd_bus_get_close_on_exit\(3\)**](#) — Control whether to close the bus connection during the event loop exit phase
[**sd_bus_get_connected_signal\(3\)**](#) — Control emission of local connection establishment signal on bus connections
[**sd_bus_get_creds_mask\(3\)**](#) — Control feature negotiation on bus connections
[**sd_bus_get_current_handler\(3\)**](#) — Query information of the callback a bus object is currently running
[**sd_bus_get_current_message\(3\)**](#) — Query information of the callback a bus object is currently running
[**sd_bus_get_current_slot\(3\)**](#) — Query information of the callback a bus object is currently running
[**sd_bus_get_current_userdata\(3\)**](#) — Query information of the callback a bus object is currently running
[**sd_bus_get_description\(3\)**](#) — Set or query properties of a bus object
[**sd_bus_get_event\(3\)**](#) — Attach a bus connection object to an event loop
[**sd_bus_get_events\(3\)**](#) — Get the file descriptor, I/O events and timeout to wait for from a message bus object
[**sd_bus_get_exit_on_disconnect\(3\)**](#) — Control the exit behavior when the bus object disconnects
[**sd_bus_get_fd\(3\)**](#) — Get the file descriptor, I/O events and timeout to wait for from a message bus object
[**sd_bus_get_method_call_timeout\(3\)**](#) — Set or query the default D-Bus method call timeout of a bus object
[**sd_bus_get_n_queued_read\(3\)**](#) — Get the number of pending bus messages in the read and write queues of a bus connection object
[**sd_bus_get_n_queued_write\(3\)**](#) — Get the number of pending bus messages in the read and write queues of a bus connection object
[**sd_bus_get_name_creds\(3\)**](#) — Query bus client credentials
[**sd_bus_get_name_machine_id\(3\)**](#) — Retrieve a bus client's machine identity
[**sd_bus_get_owner_creds\(3\)**](#) — Query bus client credentials
[**sd_bus_get_property\(3\)**](#) — Set or query D-Bus service properties
[**sd_bus_get_property_string\(3\)**](#) — Set or query D-Bus service properties
[**sd_bus_get_property_strv\(3\)**](#) — Set or query D-Bus service properties
[**sd_bus_get_property_trivial\(3\)**](#) — Set or query D-Bus service properties
[**sd_bus_get_scope\(3\)**](#) — Set or query properties of a bus object
[**sd_bus_get_sender\(3\)**](#) — Configure default sender for outgoing messages
[**sd_bus_get_tid\(3\)**](#) — Set or query properties of a bus object
[**sd_bus_get_timeout\(3\)**](#) — Get the file descriptor, I/O events and timeout to wait for from a message bus object



[`sd_bus_get_unique_name\(3\)`](#) — Set or query properties of a bus object
[`sd_bus_get_watch_bind\(3\)`](#) — Control socket binding watching on bus connections
[`sd_bus_interface_name_is_valid\(3\)`](#) — Check if a string is a valid bus name or object path
[`sd_bus_is_anonymous\(3\)`](#) — Set or query properties of a bus object
[`sd_bus_is_bus_client\(3\)`](#) — Configure connection mode for a bus object
[`sd_bus_is_monitor\(3\)`](#) — Configure connection mode for a bus object
[`sd_bus_is_open\(3\)`](#) — Check whether the bus connection is open or ready
[`sd_bus_is_ready\(3\)`](#) — Check whether the bus connection is open or ready
[`sd_bus_is_server\(3\)`](#) — Configure connection mode for a bus object
[`sd_bus_is_trusted\(3\)`](#) — Set or query properties of a bus object
[`sd_bus_list_names\(3\)`](#) — Retrieve information about registered names on a bus
[`sd_bus_match_signal\(3\)`](#) — Add a match rule for incoming message dispatching
[`sd_bus_match_signal_async\(3\)`](#) — Add a match rule for incoming message dispatching
[`sd_bus_member_name_is_valid\(3\)`](#) — Check if a string is a valid bus name or object path
[`sd_bus_message_append\(3\)`](#) — Attach fields to a D-Bus message based on a type string
[`sd_bus_message_append_array\(3\)`](#) — Append an array of fields to a D-Bus message
[`sd_bus_message_append_array_ivec\(3\)`](#) — Append an array of fields to a D-Bus message
[`sd_bus_message_append_array_memfd\(3\)`](#) — Append an array of fields to a D-Bus message
[`sd_bus_message_append_array_space\(3\)`](#) — Append an array of fields to a D-Bus message
[`sd_bus_message_append_basic\(3\)`](#) — Attach a single field to a message
[`sd_bus_message_append_string_ivec\(3\)`](#) — Attach a string to a message
[`sd_bus_message_append_string_memfd\(3\)`](#) — Attach a string to a message
[`sd_bus_message_append_string_space\(3\)`](#) — Attach a string to a message
[`sd_bus_message_append_strv\(3\)`](#) — Attach an array of strings to a message
[`sd_bus_message_appendv\(3\)`](#) — Attach fields to a D-Bus message based on a type string
[`sd_bus_message_at_end\(3\)`](#) — Check if a message has been fully read
[`sd_bus_message_close_container\(3\)`](#) — Create and move between containers in D-Bus messages
[`sd_bus_message_copy\(3\)`](#) — Copy the contents of one message to another
[`sd_bus_message_dump\(3\)`](#) — Produce a string representation of a message for debugging purposes
[`sd_bus_message_enter_container\(3\)`](#) — Create and move between containers in D-Bus messages
[`sd_bus_message_exit_container\(3\)`](#) — Create and move between containers in D-Bus messages
[`sd_bus_message_get_allow_interactive_authorization\(3\)`](#) — Set and query bus message metadata

[`sd_bus_message_get_auto_start\(3\)`](#) — Set and query bus message metadata
[`sd_bus_message_get_bus\(3\)`](#) — Create a new bus message object and create or destroy references to it
[`sd_bus_message_get_cookie\(3\)`](#) — Returns the transaction cookie of a message
[`sd_bus_message_get_creds\(3\)`](#) — Query bus message addressing/credentials metadata
[`sd_bus_message_get_destination\(3\)`](#) — Set and query bus message addressing

information

[`sd_bus_message_get_errno\(3\)`](#) — Query bus message addressing/credentials metadata
[`sd_bus_message_get_error\(3\)`](#) — Query bus message addressing/credentials metadata
[`sd_bus_message_get_expect_reply\(3\)`](#) — Set and query bus message metadata
[`sd_bus_message_get_interface\(3\)`](#) — Set and query bus message addressing information
[`sd_bus_message_get_member\(3\)`](#) — Set and query bus message addressing information
[`sd_bus_message_get_monotonic_usec\(3\)`](#) — Retrieve the sender timestamps and sequence number of a message
[`sd_bus_message_get_path\(3\)`](#) — Set and query bus message addressing information
[`sd_bus_message_get_realtime_usec\(3\)`](#) — Retrieve the sender timestamps and sequence number of a message
[`sd_bus_message_get_reply_cookie\(3\)`](#) — Returns the transaction cookie of a message
[`sd_bus_message_get_sender\(3\)`](#) — Set and query bus message addressing information
[`sd_bus_message_get_seqnum\(3\)`](#) — Retrieve the sender timestamps and sequence number of a message
[`sd_bus_message_get_signature\(3\)`](#) — Query bus message signature
[`sd_bus_message_get_type\(3\)`](#) — Query bus message addressing/credentials metadata
[`sd_bus_message_has_signature\(3\)`](#) — Query bus message signature
[`sd_bus_message_is_empty\(3\)`](#) — Query bus message signature
[`sd_bus_message_is_method_call\(3\)`](#) — Query bus message addressing/credentials metadata
[`sd_bus_message_is_method_error\(3\)`](#) — Query bus message addressing/credentials metadata
[`sd_bus_message_is_signal\(3\)`](#) — Query bus message addressing/credentials metadata
[`SD_BUS_MESSAGE_METHOD_CALL\(3\)`](#) — Create a new bus message object and create or destroy references to it
[`SD_BUS_MESSAGE_METHOD_ERROR\(3\)`](#) — Create a new bus message object and create or destroy references to it
[`SD_BUS_MESSAGE_METHOD_RETURN\(3\)`](#) — Create a new bus message object and create or destroy references to it
[`sd_bus_message_new\(3\)`](#) — Create a new bus message object and create or destroy references to it
[`sd_bus_message_new_method_call\(3\)`](#) — Create a method call message
[`sd_bus_message_new_method_errno\(3\)`](#) — Create an error reply for a method call
[`sd_bus_message_new_method_errnof\(3\)`](#) — Create an error reply for a method call
[`sd_bus_message_new_method_error\(3\)`](#) — Create an error reply for a method call
[`sd_bus_message_new_method_errorf\(3\)`](#) — Create an error reply for a method call
[`sd_bus_message_new_method_return\(3\)`](#) — Create a method call message
[`sd_bus_message_new_signal\(3\)`](#) — Create a signal message
[`sd_bus_message_new_signal_to\(3\)`](#) — Create a signal message
[`sd_bus_message_open_container\(3\)`](#) — Create and move between containers in D-Bus messages
[`sd_bus_message_peek_type\(3\)`](#) — Read a sequence of values from a message
[`sd_bus_message_read\(3\)`](#) — Read a sequence of values from a message
[`sd_bus_message_read_array\(3\)`](#) — Access an array of elements in a message
[`sd_bus_message_read_basic\(3\)`](#) — Read a basic type from a message
[`sd_bus_message_read_strv\(3\)`](#) — Access an array of strings in a message



[**sd_bus_message_read_strv_extend\(3\)**](#) — Access an array of strings in a message
[**sd_bus_message_readv\(3\)**](#) — Read a sequence of values from a message
[**sd_bus_message_ref\(3\)**](#) — Create a new bus message object and create or destroy references to it
[**sd_bus_message_rewind\(3\)**](#) — Return to beginning of message or current container
[**sd_bus_message_seal\(3\)**](#) — Prepare a D-Bus message for transmission
[**sd_bus_message_send\(3\)**](#) — Queue a D-Bus message for transfer
[**sd_bus_message_sensitive\(3\)**](#) — Mark a message object as containing sensitive data
[**sd_bus_message_set_allow_interactive_authorization\(3\)**](#) — Set and query bus message metadata
[**sd_bus_message_set_auto_start\(3\)**](#) — Set and query bus message metadata
[**sd_bus_message_set_destination\(3\)**](#) — Set and query bus message addressing information
[**sd_bus_message_set_expect_reply\(3\)**](#) — Set and query bus message metadata
[**sd_bus_message_set_sender\(3\)**](#) — Set and query bus message addressing information
[**SD_BUS_MESSAGE_SIGNAL\(3\)**](#) — Create a new bus message object and create or destroy references to it
[**sd_bus_message_skip\(3\)**](#) — Skip elements in a bus message
[**sd_bus_message_unref\(3\)**](#) — Create a new bus message object and create or destroy references to it
[**sd_bus_message_unrefp\(3\)**](#) — Create a new bus message object and create or destroy references to it
[**sd_bus_message_verify_type\(3\)**](#) — Check if the message has specified type at the current location
[**SD_BUS_METHOD\(3\)**](#) — Declare properties and methods for a D-Bus path
[**SD_BUS_METHOD_WITH_NAMES\(3\)**](#) — Declare properties and methods for a D-Bus path
[**SD_BUS_METHOD_WITH_NAMES_OFFSET\(3\)**](#) — Declare properties and methods for a D-Bus path
[**SD_BUS_METHOD_WITH_OFFSET\(3\)**](#) — Declare properties and methods for a D-Bus path
[**sd_bus_negotiate_creds\(3\)**](#) — Control feature negotiation on bus connections
[**sd_bus_negotiate_fds\(3\)**](#) — Control feature negotiation on bus connections
[**sd_bus_negotiate_timestamp\(3\)**](#) — Control feature negotiation on bus connections
[**sd_bus_new\(3\)**](#) — Create a new bus object and create or destroy references to it
[**sd_bus_object_path_is_valid\(3\)**](#) — Check if a string is a valid bus name or object path
[**sd_bus_open\(3\)**](#) — Acquire a connection to a system or user bus
[**sd_bus_open_system\(3\)**](#) — Acquire a connection to a system or user bus
[**sd_bus_open_system_machine\(3\)**](#) — Acquire a connection to a system or user bus
[**sd_bus_open_system_remote\(3\)**](#) — Acquire a connection to a system or user bus
[**sd_bus_open_system_with_description\(3\)**](#) — Acquire a connection to a system or user bus
[**sd_bus_open_user\(3\)**](#) — Acquire a connection to a system or user bus
[**sd_bus_open_user_machine\(3\)**](#) — Acquire a connection to a system or user bus
[**sd_bus_open_user_with_description\(3\)**](#) — Acquire a connection to a system or user bus
[**sd_bus_open_with_description\(3\)**](#) — Acquire a connection to a system or user bus
[**SD_BUS_PARAM\(3\)**](#) — Declare properties and methods for a D-Bus path
[**sd_bus_path_decode\(3\)**](#) — Convert an external identifier into an object path and back



[`sd_bus_path_decode_many\(3\)`](#) — Convert an external identifier into an object path and back
[`sd_bus_path_encode\(3\)`](#) — Convert an external identifier into an object path and back
[`sd_bus_path_encode_many\(3\)`](#) — Convert an external identifier into an object path and back
[`sd_bus_process\(3\)`](#) — Drive the connection
[`SD_BUS_PROPERTY\(3\)`](#) — Declare properties and methods for a D-Bus path
[`sd_bus_query_sender_creds\(3\)`](#) — Query bus message sender credentials/privileges
[`sd_bus_query_sender_privilege\(3\)`](#) — Query bus message sender credentials/privileges
[`sd_bus_ref\(3\)`](#) — Create a new bus object and create or destroy references to it
[`sd_bus_release_name\(3\)`](#) — Request or release a well-known service name on a bus
[`sd_bus_release_name_async\(3\)`](#) — Request or release a well-known service name on a bus
[`sd_bus_reply_method_errno\(3\)`](#) — Reply with an error to a D-Bus method call
[`sd_bus_reply_method_errnof\(3\)`](#) — Reply with an error to a D-Bus method call
[`sd_bus_reply_method_errnofv\(3\)`](#) — Reply with an error to a D-Bus method call
[`sd_bus_reply_method_error\(3\)`](#) — Reply with an error to a D-Bus method call
[`sd_bus_reply_method_errorf\(3\)`](#) — Reply with an error to a D-Bus method call
[`sd_bus_reply_method_errorfv\(3\)`](#) — Reply with an error to a D-Bus method call
[`sd_bus_reply_method_return\(3\)`](#) — Reply to a D-Bus method call
[`sd_bus_reply_method_returnv\(3\)`](#) — Reply to a D-Bus method call
[`sd_bus_request_name\(3\)`](#) — Request or release a well-known service name on a bus
[`sd_bus_request_name_async\(3\)`](#) — Request or release a well-known service name on a bus
[`sd_bus_send\(3\)`](#) — Queue a D-Bus message for transfer
[`sd_bus_send_to\(3\)`](#) — Queue a D-Bus message for transfer
[`sd_bus_service_name_is_valid\(3\)`](#) — Check if a string is a valid bus name or object path
[`sd_bus_set_address\(3\)`](#) — Set or query the address of the bus connection
[`sd_bus_set_allow_interactive_authorization\(3\)`](#) — Set or query properties of a bus object
[`sd_bus_set_anonymous\(3\)`](#) — Set or query properties of a bus object
[`sd_bus_set_bus_client\(3\)`](#) — Configure connection mode for a bus object
[`sd_bus_set_close_on_exit\(3\)`](#) — Control whether to close the bus connection during the event loop exit phase
[`sd_bus_set_connected_signal\(3\)`](#) — Control emission of local connection establishment signal on bus connections
[`sd_bus_set_description\(3\)`](#) — Set or query properties of a bus object
[`sd_bus_set_exec\(3\)`](#) — Set or query the address of the bus connection
[`sd_bus_set_exit_on_disconnect\(3\)`](#) — Control the exit behavior when the bus object disconnects
[`sd_bus_set_fd\(3\)`](#) — Set the file descriptors to use for bus communication

[`sd_bus_set_method_call_timeout\(3\)`](#) — Set or query the default D-Bus method call timeout of a bus object
[`sd_bus_set_monitor\(3\)`](#) — Configure connection mode for a bus object
[`sd_bus_set_property\(3\)`](#) — Set or query D-Bus service properties
[`sd_bus_set_propertyv\(3\)`](#) — Set or query D-Bus service properties
[`sd_bus_set_sender\(3\)`](#) — Configure default sender for outgoing messages
[`sd_bus_set_server\(3\)`](#) — Configure connection mode for a bus object

[`sd_bus_set_trusted\(3\)`](#) — Set or query properties of a bus object
[`sd_bus_set_watch_bind\(3\)`](#) — Control socket binding watching on bus connections
[`SD_BUS_SIGNAL\(3\)`](#) — Declare properties and methods for a D-Bus path
[`SD_BUS_SIGNAL_WITH_NAMES\(3\)`](#) — Declare properties and methods for a D-Bus path
[`sd_bus_slot_get_bus\(3\)`](#) — Query information attached to a bus slot object
[`sd_bus_slot_get_current_handler\(3\)`](#) — Query information attached to a bus slot object
[`sd_bus_slot_get_current_message\(3\)`](#) — Query information attached to a bus slot object
[`sd_bus_slot_get_current_userdata\(3\)`](#) — Query information attached to a bus slot object
[`sd_bus_slot_get_description\(3\)`](#) — Set or query the description of bus slot objects
[`sd_bus_slot_get_destroy_callback\(3\)`](#) — Define the callback function for resource cleanup
[`sd_bus_slot_get_floating\(3\)`](#) — Control whether a bus slot object is "floating"
[`sd_bus_slot_get_userdata\(3\)`](#) — Set and query the value in the "userdata" field
[`sd_bus_slot_ref\(3\)`](#) — Create and destroy references to a bus slot object
[`sd_bus_slot_set_description\(3\)`](#) — Set or query the description of bus slot objects
[`sd_bus_slot_set_destroy_callback\(3\)`](#) — Define the callback function for resource cleanup
[`sd_bus_slot_set_floating\(3\)`](#) — Control whether a bus slot object is "floating"
[`sd_bus_slot_set_userdata\(3\)`](#) — Set and query the value in the "userdata" field
[`sd_bus_slot_unref\(3\)`](#) — Create and destroy references to a bus slot object
[`sd_bus_slot_unrefp\(3\)`](#) — Create and destroy references to a bus slot object
[`sd_bus_start\(3\)`](#) — Initiate a bus connection to the D-bus broker daemon
[`sd_bus_track_add_name\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object
[`sd_bus_track_add_sender\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object
[`sd_bus_track_contains\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object
[`sd_bus_track_count\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object
[`sd_bus_track_count_name\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object
[`sd_bus_track_count_sender\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object
[`sd_bus_track_first\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object
[`sd_bus_track_get_bus\(3\)`](#) — Track bus peers
[`sd_bus_track_get_destroy_callback\(3\)`](#) — Define the callback function for resource cleanup 
[`sd_bus_track_get_recursive\(3\)`](#) — Track bus peers
[`sd_bus_track_get_userdata\(3\)`](#) — Track bus peers
[`sd_bus_track_new\(3\)`](#) — Track bus peers
[`sd_bus_track_next\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object
[`sd_bus_track_ref\(3\)`](#) — Track bus peers
[`sd_bus_track_remove_name\(3\)`](#) — Add, remove and retrieve bus peers tracked in a bus

peer tracking object

[sd_bus_track_remove_sender\(3\)](#) — Add, remove and retrieve bus peers tracked in a bus peer tracking object

[sd_bus_track_set_destroy_callback\(3\)](#) — Define the callback function for resource cleanup

[sd_bus_track_set_recursive\(3\)](#) — Track bus peers

[sd_bus_track_set_userdata\(3\)](#) — Track bus peers

[sd_bus_track_unref\(3\)](#) — Track bus peers

[sd_bus_track_unrefp\(3\)](#) — Track bus peers

[sd_bus_unref\(3\)](#) — Create a new bus object and create or destroy references to it

[sd_bus_unrefp\(3\)](#) — Create a new bus object and create or destroy references to it

[SD_BUS_VTABLE_CAPABILITY\(3\)](#) — Declare properties and methods for a D-Bus path

[SD_BUS_VTABLE_END\(3\)](#) — Declare properties and methods for a D-Bus path

[SD_BUS_VTABLE_START\(3\)](#) — Declare properties and methods for a D-Bus path

[sd_bus_wait\(3\)](#) — Wait for I/O on a bus connection

[SD_BUS_WRITABLE_PROPERTY\(3\)](#) — Declare properties and methods for a D-Bus path

[SD_CRIT\(3\)](#) — APIs for new-style daemons

[SD_DEBUG\(3\)](#) — APIs for new-style daemons

[sd_device_get_devname\(3\)](#) — Returns various fields of device objects

[sd_device_get_devnum\(3\)](#) — Returns various fields of device objects

[sd_device_get_devpath\(3\)](#) — Returns various fields of device objects

[sd_device_get_devtype\(3\)](#) — Returns various fields of device objects

[sd_device_get_diskseq\(3\)](#) — Returns various fields of device objects

[sd_device_get_driver\(3\)](#) — Returns various fields of device objects

[sd_device_get_ifindex\(3\)](#) — Returns various fields of device objects

[sd_device_get_subsystem\(3\)](#) — Returns various fields of device objects

[sd_device_get_sysname\(3\)](#) — Returns various fields of device objects

[sd_device_get_sysnum\(3\)](#) — Returns various fields of device objects

[sd_device_get_syspath\(3\)](#) — Returns various fields of device objects

[sd_device_ref\(3\)](#) — Create or destroy references to a device object

[sd_device_unref\(3\)](#) — Create or destroy references to a device object

[sd_device_unrefp\(3\)](#) — Create or destroy references to a device object

[SD_EMERG\(3\)](#) — APIs for new-style daemons

[SD_ERR\(3\)](#) — APIs for new-style daemons

[sd_event\(3\)](#) — Acquire and release an event loop object

[sd_event_add_child\(3\)](#) — Add a child process state change event source to an event loop

[sd_event_add_child_pidfd\(3\)](#) — Add a child process state change event source to an event loop

[sd_event_add_defer\(3\)](#) — Add static event sources to an event loop

[sd_event_add_exit\(3\)](#) — Add static event sources to an event loop

[sd_event_add_inotify\(3\)](#) — Add an "inotify" file system inode event source to an event loop

[sd_event_add_inotify_fd\(3\)](#) — Add an "inotify" file system inode event source to an event loop

[sd_event_add_io\(3\)](#) — Add an I/O event source to an event loop

[sd_event_add_memory_pressure\(3\)](#) — Add and configure an event source run as result of memory pressure



[`sd_event_add_post\(3\)`](#) — Add static event sources to an event loop
[`sd_event_add_signal\(3\)`](#) — Add a UNIX process signal event source to an event loop
[`sd_event_add_time\(3\)`](#) — Add a timer event source to an event loop
[`sd_event_add_time_relative\(3\)`](#) — Add a timer event source to an event loop
[`SD_EVENT_ARMED\(3\)`](#) — Low-level event loop operations
[`sd_event_child_handler_t\(3\)`](#) — Add a child process state change event source to an event loop
[`sd_event_default\(3\)`](#) — Acquire and release an event loop object
[`sd_event_destroy_t\(3\)`](#) — Define the callback function for resource cleanup
[`sd_event_dispatch\(3\)`](#) — Low-level event loop operations
[`sd_event_exit\(3\)`](#) — Ask the event loop to exit
[`SD_EVENT_EXITING\(3\)`](#) — Low-level event loop operations
[`SD_EVENT_FINISHED\(3\)`](#) — Low-level event loop operations
[`sd_event_get_exit_code\(3\)`](#) — Ask the event loop to exit
[`sd_event_get_fd\(3\)`](#) — Obtain a file descriptor to poll for event loop events
[`sd_event_get_iteration\(3\)`](#) — Low-level event loop operations
[`sd_event_get_state\(3\)`](#) — Low-level event loop operations
[`sd_event_get_tid\(3\)`](#) — Acquire and release an event loop object
[`sd_event_get_watchdog\(3\)`](#) — Enable event loop watchdog support
[`sd_event_handler_t\(3\)`](#) — Add static event sources to an event loop
[`SD_EVENT_INITIAL\(3\)`](#) — Low-level event loop operations
[`sd_event_inotify_handler_t\(3\)`](#) — Add an "inotify" file system inode event source to an event loop
[`sd_event_io_handler_t\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_loop\(3\)`](#) — Run an event loop
[`sd_event_new\(3\)`](#) — Acquire and release an event loop object
[`sd_event_now\(3\)`](#) — Retrieve current event loop iteration timestamp
[`SD_EVENT_OFF\(3\)`](#) — Enable or disable event sources
[`SD_EVENT_ON\(3\)`](#) — Enable or disable event sources
[`SD_EVENT_ONESHOT\(3\)`](#) — Enable or disable event sources
[`SD_EVENT_PENDING\(3\)`](#) — Low-level event loop operations
[`sd_event_prepare\(3\)`](#) — Low-level event loop operations
[`SD_EVENT_PREPARING\(3\)`](#) — Low-level event loop operations
[`SD_EVENT_PRIORITY_IDLE\(3\)`](#) — Set or retrieve the priority of event sources
[`SD_EVENT_PRIORITY_IMPORTANT\(3\)`](#) — Set or retrieve the priority of event sources
[`SD_EVENT_PRIORITY_NORMAL\(3\)`](#) — Set or retrieve the priority of event sources
[`sd_event_ref\(3\)`](#) — Acquire and release an event loop object
[`sd_event_run\(3\)`](#) — Run an event loop
[`SD_EVENT_RUNNING\(3\)`](#) — Low-level event loop operations
[`sd_event_set_signal_exit\(3\)`](#) — Automatically leave event loop on SIGINT and SIGTERM

[`sd_event_set_watchdog\(3\)`](#) — Enable event loop watchdog support
[`sd_event_signal_handler_t\(3\)`](#) — Add a UNIX process signal event source to an event loop
[`SD_EVENT_SIGNAL_PROCMASK\(3\)`](#) — Add a UNIX process signal event source to an event loop
[`sd_event_source\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_source_disable_unref\(3\)`](#) — Increase or decrease event source reference counters

[`sd_event_source_disable_unrefp\(3\)`](#) — Increase or decrease event source reference counters
[`sd_event_source_get_child_pid\(3\)`](#) — Add a child process state change event source to an event loop
[`sd_event_source_get_child_pidfd\(3\)`](#) — Add a child process state change event source to an event loop
[`sd_event_source_get_child_pidfd_own\(3\)`](#) — Add a child process state change event source to an event loop
[`sd_event_source_get_child_process_own\(3\)`](#) — Add a child process state change event source to an event loop
[`sd_event_source_get_description\(3\)`](#) — Set or retrieve descriptive names of event sources
[`sd_event_source_get_destroy_callback\(3\)`](#) — Define the callback function for resource cleanup
[`sd_event_source_get_enabled\(3\)`](#) — Enable or disable event sources
[`sd_event_source_get_event\(3\)`](#) — Retrieve the event loop of an event source
[`sd_event_source_get_exit_on_failure\(3\)`](#) — Set or retrieve the exit-on-failure feature of event sources
[`sd_event_source_get_floating\(3\)`](#) — Set or retrieve 'floating' state of event sources
[`sd_event_source_get_inotify_mask\(3\)`](#) — Add an "inotify" file system inode event source to an event loop
[`sd_event_source_get_io_events\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_source_get_io_fd\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_source_get_io_fd_own\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_source_get_io_revents\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_source_get_pending\(3\)`](#) — Determine pending state of event sources
[`sd_event_source_get_priority\(3\)`](#) — Set or retrieve the priority of event sources
[`sd_event_source_get_ratelimit\(3\)`](#) — Configure rate limiting on event sources
[`sd_event_source_get_signal\(3\)`](#) — Add a UNIX process signal event source to an event loop
[`sd_event_source_get_time\(3\)`](#) — Add a timer event source to an event loop
[`sd_event_source_get_time_accuracy\(3\)`](#) — Add a timer event source to an event loop
[`sd_event_source_get_time_clock\(3\)`](#) — Add a timer event source to an event loop
[`sd_event_source_get_userdata\(3\)`](#) — Set or retrieve user data pointer of event sources
[`sd_event_source_is_ratelimited\(3\)`](#) — Configure rate limiting on event sources
[`sd_event_source_leave_ratelimit\(3\)`](#) — Configure rate limiting on event sources
[`sd_event_source_ref\(3\)`](#) — Increase or decrease event source reference counters
[`sd_event_source_send_child_signal\(3\)`](#) — Add a child process state change event source to an event loop
[`sd_event_source_set_child_pidfd_own\(3\)`](#) — Add a child process state change event source to an event loop 
[`sd_event_source_set_child_process_own\(3\)`](#) — Add a child process state change event source to an event loop
[`sd_event_source_set_description\(3\)`](#) — Set or retrieve descriptive names of event sources
[`sd_event_source_set_destroy_callback\(3\)`](#) — Define the callback function for resource cleanup
[`sd_event_source_set_enabled\(3\)`](#) — Enable or disable event sources
[`sd_event_source_set_exit_on_failure\(3\)`](#) — Set or retrieve the exit-on-failure feature of

event sources

[`sd_event_source_set_floating\(3\)`](#) — Set or retrieve 'floating' state of event sources
[`sd_event_source_set_io_events\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_source_set_io_fd\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_source_set_io_fd_own\(3\)`](#) — Add an I/O event source to an event loop
[`sd_event_source_set_memory_pressure_period\(3\)`](#) — Add and configure an event source run as result of memory pressure
[`sd_event_source_set_memory_pressure_type\(3\)`](#) — Add and configure an event source run as result of memory pressure
[`sd_event_source_set_prepare\(3\)`](#) — Set a preparation callback for event sources
[`sd_event_source_set_priority\(3\)`](#) — Set or retrieve the priority of event sources
[`sd_event_source_set_ratelimit\(3\)`](#) — Configure rate limiting on event sources
[`sd_event_source_set_ratelimit_expire_callback\(3\)`](#) — Configure rate limiting on event sources
[`sd_event_source_set_time\(3\)`](#) — Add a timer event source to an event loop
[`sd_event_source_set_time_accuracy\(3\)`](#) — Add a timer event source to an event loop
[`sd_event_source_set_time_relative\(3\)`](#) — Add a timer event source to an event loop
[`sd_event_source_set_userdata\(3\)`](#) — Set or retrieve user data pointer of event sources
[`sd_event_source_unref\(3\)`](#) — Increase or decrease event source reference counters
[`sd_event_source_unrefp\(3\)`](#) — Increase or decrease event source reference counters
[`sd_event_time_handler_t\(3\)`](#) — Add a timer event source to an event loop
[`sd_event_trim_memory\(3\)`](#) — Add and configure an event source run as result of memory pressure
[`sd_event_unref\(3\)`](#) — Acquire and release an event loop object
[`sd_event_unrefp\(3\)`](#) — Acquire and release an event loop object
[`sd_event_wait\(3\)`](#) — Low-level event loop operations
[`sd_get_machine_names\(3\)`](#) — Determine available seats, sessions, logged in users and virtual machines/containers
[`sd_get_seats\(3\)`](#) — Determine available seats, sessions, logged in users and virtual machines/containers
[`sd_get_sessions\(3\)`](#) — Determine available seats, sessions, logged in users and virtual machines/containers
[`sd_get_uids\(3\)`](#) — Determine available seats, sessions, logged in users and virtual machines/containers
[`sd_hwdb_enumerate\(3\)`](#) — Seek to a location in hwdb or access entries
[`SD_HWDB_FOREACH_PROPERTY\(3\)`](#) — Seek to a location in hwdb or access entries
[`sd_hwdb_get\(3\)`](#) — Seek to a location in hwdb or access entries
[`sd_hwdb_new\(3\)`](#) — Create a new hwdb object and create or destroy references to it
[`sd_hwdb_new_from_path\(3\)`](#) — Create a new hwdb object and create or destroy references to it
[`sd_hwdb_ref\(3\)`](#) — Create a new hwdb object and create or destroy references to it 
[`sd_hwdb_seek\(3\)`](#) — Seek to a location in hwdb or access entries
[`sd_hwdb_unref\(3\)`](#) — Create a new hwdb object and create or destroy references to it
[`SD_ID128_ALLF\(3\)`](#) — APIs for processing 128-bit IDs
[`SD_ID128_CONST_STR\(3\)`](#) — APIs for processing 128-bit IDs
[`sd_id128_equal\(3\)`](#) — APIs for processing 128-bit IDs
[`SD_ID128_FORMAT_STR\(3\)`](#) — APIs for processing 128-bit IDs
[`SD_ID128_FORMAT_VAL\(3\)`](#) — APIs for processing 128-bit IDs
[`sd_id128_from_string\(3\)`](#) — Format or parse 128-bit IDs as strings

[sd_id128_get_app_specific\(3\)](#) — Retrieve 128-bit IDs
[sd_id128_get_boot\(3\)](#) — Retrieve 128-bit IDs
[sd_id128_get_boot_app_specific\(3\)](#) — Retrieve 128-bit IDs
[sd_id128_get_invocation\(3\)](#) — Retrieve 128-bit IDs
[sd_id128_get_machine\(3\)](#) — Retrieve 128-bit IDs
[sd_id128_get_machine_app_specific\(3\)](#) — Retrieve 128-bit IDs
[sd_id128_in_set\(3\)](#) — APIs for processing 128-bit IDs
[sd_id128_in_set_sentinel\(3\)](#) — APIs for processing 128-bit IDs
[sd_id128_in_setv\(3\)](#) — APIs for processing 128-bit IDs
[sd_id128_is_allf\(3\)](#) — APIs for processing 128-bit IDs
[sd_id128_is_null\(3\)](#) — APIs for processing 128-bit IDs
[SD_ID128_MAKE\(3\)](#) — APIs for processing 128-bit IDs
[SD_ID128_MAKE_STR\(3\)](#) — APIs for processing 128-bit IDs
[SD_ID128_MAKE_UUID_STR\(3\)](#) — APIs for processing 128-bit IDs
[SD_ID128_NULL\(3\)](#) — APIs for processing 128-bit IDs
[sd_id128_randomize\(3\)](#) — Generate 128-bit IDs
[sd_id128_string_equal\(3\)](#) — APIs for processing 128-bit IDs
[SD_ID128_STRING_MAX\(3\)](#) — Format or parse 128-bit IDs as strings
[sd_id128_t\(3\)](#) — APIs for processing 128-bit IDs
[sd_id128_to_string\(3\)](#) — Format or parse 128-bit IDs as strings
[SD_ID128_TO_STRING\(3\)](#) — Format or parse 128-bit IDs as strings
[sd_id128_to_uuid_string\(3\)](#) — Format or parse 128-bit IDs as strings
[SD_ID128_TO_UUID_STRING\(3\)](#) — Format or parse 128-bit IDs as strings
[SD_ID128_UUID_FORMAT_STR\(3\)](#) — APIs for processing 128-bit IDs
[SD_ID128_UUID_STRING_MAX\(3\)](#) — Format or parse 128-bit IDs as strings
[SD_INFO\(3\)](#) — APIs for new-style daemons
[sd_is_fifo\(3\)](#) — Check the type of a file descriptor
[sd_is_mq\(3\)](#) — Check the type of a file descriptor
[sd_is_socket\(3\)](#) — Check the type of a file descriptor
[sd_is_socket_inet\(3\)](#) — Check the type of a file descriptor
[sd_is_socket_sockaddr\(3\)](#) — Check the type of a file descriptor
[sd_is_socket_unix\(3\)](#) — Check the type of a file descriptor
[sd_is_special\(3\)](#) — Check the type of a file descriptor
[sd_journal\(3\)](#) — Open the system journal for reading
[sd_journal_add_conjunction\(3\)](#) — Add or remove entry matches
[sd_journal_add_disjunction\(3\)](#) — Add or remove entry matches
[sd_journal_add_match\(3\)](#) — Add or remove entry matches
[SD_JOURNAL_ALL_NAMESPACES\(3\)](#) — Open the system journal for reading
[SD_JOURNAL_APPEND\(3\)](#) — Journal change notification interface
[sd_journal_close\(3\)](#) — Open the system journal for reading
[SD_JOURNAL_CURRENT_USER\(3\)](#) — Open the system journal for reading

[sd_journal_enumerate_available_data\(3\)](#) — Read data fields from the current journal entry
[sd_journal_enumerate_available_unique\(3\)](#) — Read unique data fields from the journal
[sd_journal_enumerate_data\(3\)](#) — Read data fields from the current journal entry
[sd_journal_enumerate_fields\(3\)](#) — Read used field names from the journal
[sd_journal_enumerate_unique\(3\)](#) — Read unique data fields from the journal
[sd_journal_flush_matches\(3\)](#) — Add or remove entry matches
[SD_JOURNAL_FOREACH\(3\)](#) — Advance or set back the read pointer in the journal

[**SD_JOURNAL_FOREACH_BACKWARDS\(3\)**](#) — Advance or set back the read pointer in the journal

[**SD_JOURNAL_FOREACH_DATA\(3\)**](#) — Read data fields from the current journal entry

[**SD_JOURNAL_FOREACH_FIELD\(3\)**](#) — Read used field names from the journal

[**SD_JOURNAL_FOREACH_UNIQUE\(3\)**](#) — Read unique data fields from the journal

[**sd_journal_get_catalog\(3\)**](#) — Retrieve message catalog entry

[**sd_journal_get_catalog_for_message_id\(3\)**](#) — Retrieve message catalog entry

[**sd_journal_get_cursor\(3\)**](#) — Get cursor string for or test cursor string against the current journal entry

[**sd_journal_get_cutoff_monotonic_usec\(3\)**](#) — Read cut-off timestamps from the current journal entry

[**sd_journal_get_cutoff_realtime_usec\(3\)**](#) — Read cut-off timestamps from the current journal entry

[**sd_journal_get_data\(3\)**](#) — Read data fields from the current journal entry

[**sd_journal_get_data_threshold\(3\)**](#) — Read data fields from the current journal entry

[**sd_journal_get_events\(3\)**](#) — Journal change notification interface

[**sd_journal_get_fd\(3\)**](#) — Journal change notification interface

[**sd_journal_get_monotonic_usec\(3\)**](#) — Read timestamps from the current journal entry

[**sd_journal_get_realtime_usec\(3\)**](#) — Read timestamps from the current journal entry

[**sd_journal_get_seqnum\(3\)**](#) — Read sequence number from the current journal entry

[**sd_journal_get_timeout\(3\)**](#) — Journal change notification interface

[**sd_journal_get_usage\(3\)**](#) — Journal disk usage

[**sd_journal_has_persistent_files\(3\)**](#) — Query availability of runtime or persistent journal files

[**sd_journal_has_runtime_files\(3\)**](#) — Query availability of runtime or persistent journal files

[**SD_JOURNAL_INCLUDE_DEFAULT_NAMESPACE\(3\)**](#) — Open the system journal for reading

[**SD_JOURNAL_INVALIDATE\(3\)**](#) — Journal change notification interface

[**SD_JOURNAL_LOCAL_ONLY\(3\)**](#) — Open the system journal for reading

[**sd_journal_next\(3\)**](#) — Advance or set back the read pointer in the journal

[**sd_journal_next_skip\(3\)**](#) — Advance or set back the read pointer in the journal

[**SD_JOURNAL_NOP\(3\)**](#) — Journal change notification interface

[**sd_journal_open\(3\)**](#) — Open the system journal for reading

[**sd_journal_open_directory\(3\)**](#) — Open the system journal for reading

[**sd_journal_open_directory_fd\(3\)**](#) — Open the system journal for reading

[**sd_journal_open_files\(3\)**](#) — Open the system journal for reading

[**sd_journal_open_files_fd\(3\)**](#) — Open the system journal for reading

[**sd_journal_open_namespace\(3\)**](#) — Open the system journal for reading

[**SD_JOURNAL_OS_ROOT\(3\)**](#) — Open the system journal for reading

[**sd_journal_perror\(3\)**](#) — Submit log entries to the journal

[**sd_journal_perror_with_location\(3\)**](#) — Submit log entries to the journal

[**sd_journal_previous\(3\)**](#) — Advance or set back the read pointer in the journal

[**sd_journal_previous_skip\(3\)**](#) — Advance or set back the read pointer in the journal

[**sd_journal_print\(3\)**](#) — Submit log entries to the journal

[**sd_journal_print_with_location\(3\)**](#) — Submit log entries to the journal

[**sd_journal_printv\(3\)**](#) — Submit log entries to the journal

[**sd_journal_printv_with_location\(3\)**](#) — Submit log entries to the journal

[**sd_journal_process\(3\)**](#) — Journal change notification interface



[**sd_journal_query_unique\(3\)**](#) — Read unique data fields from the journal
[**sd_journal_reliable_fd\(3\)**](#) — Journal change notification interface
[**sd_journal_restart_data\(3\)**](#) — Read data fields from the current journal entry
[**sd_journal_restart_fields\(3\)**](#) — Read used field names from the journal
[**sd_journal_restart_unique\(3\)**](#) — Read unique data fields from the journal
[**SD_JOURNAL_RUNTIME_ONLY\(3\)**](#) — Open the system journal for reading
[**sd_journal_seek_cursor\(3\)**](#) — Seek to a position in the journal
[**sd_journal_seek_head\(3\)**](#) — Seek to a position in the journal
[**sd_journal_seek_monotonic_usec\(3\)**](#) — Seek to a position in the journal
[**sd_journal_seek_realtime_usec\(3\)**](#) — Seek to a position in the journal
[**sd_journal_seek_tail\(3\)**](#) — Seek to a position in the journal
[**sd_journal_send\(3\)**](#) — Submit log entries to the journal
[**sd_journal_send_with_location\(3\)**](#) — Submit log entries to the journal
[**sd_journal_sendv\(3\)**](#) — Submit log entries to the journal
[**sd_journal_sendv_with_location\(3\)**](#) — Submit log entries to the journal
[**sd_journal_set_data_threshold\(3\)**](#) — Read data fields from the current journal entry
[**sd_journal_step_one\(3\)**](#) — Advance or set back the read pointer in the journal
[**sd_journal_stream_fd\(3\)**](#) — Create log stream file descriptor to the journal
[**SD_JOURNAL_SUPPRESS_LOCATION\(3\)**](#) — Submit log entries to the journal
[**SD_JOURNAL_SYSTEM\(3\)**](#) — Open the system journal for reading
[**SD_JOURNAL_TAKE_DIRECTORY_FD\(3\)**](#) — Open the system journal for reading
[**sd_journal_test_cursor\(3\)**](#) — Get cursor string for or test cursor string against the current journal entry
[**sd_journal_wait\(3\)**](#) — Journal change notification interface
[**sd_listen_fds\(3\)**](#) — Check for file descriptors passed by the system manager
[**SD_LISTEN_FDS_START\(3\)**](#) — Check for file descriptors passed by the system manager
[**sd_listen_fds_with_names\(3\)**](#) — Check for file descriptors passed by the system manager
[**sd_login_monitor\(3\)**](#) — Monitor login sessions, seats, users and virtual machines/containers
[**sd_login_monitor_flush\(3\)**](#) — Monitor login sessions, seats, users and virtual machines/containers
[**sd_login_monitor_get_events\(3\)**](#) — Monitor login sessions, seats, users and virtual machines/containers
[**sd_login_monitor_get_fd\(3\)**](#) — Monitor login sessions, seats, users and virtual machines/containers
[**sd_login_monitor_get_timeout\(3\)**](#) — Monitor login sessions, seats, users and virtual machines/containers
[**sd_login_monitor_new\(3\)**](#) — Monitor login sessions, seats, users and virtual machines/containers

[**sd_login_monitor_unref\(3\)**](#) — Monitor login sessions, seats, users and virtual machines/containers
[**sd_login_monitor_unrefp\(3\)**](#) — Monitor login sessions, seats, users and virtual machines/containers
[**sd_machine_get_class\(3\)**](#) — Determine the class and network interface indices of a locally running virtual machine or container
[**sd_machine_get_ifindices\(3\)**](#) — Determine the class and network interface indices of a locally running virtual machine or container

[**SD_NOTICE\(3\)**](#) — APIs for new-style daemons

[**sd_notify\(3\)**](#) — Notify service manager about start-up completion and other service status changes

[**sd_notify_barrier\(3\)**](#) — Notify service manager about start-up completion and other service status changes

[**sd_notifyf\(3\)**](#) — Notify service manager about start-up completion and other service status changes

[**sd_path_lookup\(3\)**](#) — Query well-known file system paths

[**sd_path_lookup_strv\(3\)**](#) — Query well-known file system paths

[**sd_peer_get_cgroup\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_peer_get_machine_name\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_peer_get_owner_uid\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_peer_get_session\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_peer_get_slice\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_peer_get_unit\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_peer_get_user_slice\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_peer_get_user_unit\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_pid_get_cgroup\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_pid_get_machine_name\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_pid_get_owner_uid\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_pid_get_session\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_pid_get_slice\(3\)**](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[**sd_pid_get_unit\(3\)**](#) — Determine the owner uid of the user unit or session, or the

session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pid_get_user_slice\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pid_get_user_unit\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pid_notify\(3\)`](#) — Notify service manager about start-up completion and other service status changes

[`sd_pid_notify_barrier\(3\)`](#) — Notify service manager about start-up completion and other service status changes

[`sd_pid_notify_with_fds\(3\)`](#) — Notify service manager about start-up completion and other service status changes

[`sd_pid_notifyf\(3\)`](#) — Notify service manager about start-up completion and other service status changes

[`sd_pid_notifyf_with_fds\(3\)`](#) — Notify service manager about start-up completion and other service status changes

[`sd_pidfd_get_cgroup\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pidfd_get_machine_name\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pidfd_get_owner_uid\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pidfd_get_session\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pidfd_get_slice\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pidfd_get_unit\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pidfd_get_user_slice\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_pidfd_get_user_unit\(3\)`](#) — Determine the owner uid of the user unit or session, or the session, user unit, system unit, container/VM or slice that a specific PID or socket peer belongs to

[`sd_seat_can_graphical\(3\)`](#) — Determine state of a specific seat

[`sd_seat_can_tty\(3\)`](#) — Determine state of a specific seat

[`sd_seat_get_active\(3\)`](#) — Determine state of a specific seat

[`sd_seat_get_sessions\(3\)`](#) — Determine state of a specific seat

[`sd_session_get_class\(3\)`](#) — Determine state of a specific session

[`sd_session_get_desktop\(3\)`](#) — Determine state of a specific session

[`sd_session_get_display\(3\)`](#) — Determine state of a specific session

[**sd_session_get_leader\(3\)**](#) — Determine state of a specific session
[**sd_session_get_remote_host\(3\)**](#) — Determine state of a specific session
[**sd_session_get_remote_user\(3\)**](#) — Determine state of a specific session
[**sd_session_get_seat\(3\)**](#) — Determine state of a specific session
[**sd_session_get_service\(3\)**](#) — Determine state of a specific session
[**sd_session_get_start_time\(3\)**](#) — Determine state of a specific session
[**sd_session_get_state\(3\)**](#) — Determine state of a specific session
[**sd_session_get_tty\(3\)**](#) — Determine state of a specific session
[**sd_session_get_type\(3\)**](#) — Determine state of a specific session
[**sd_session_get_uid\(3\)**](#) — Determine state of a specific session
[**sd_session_get_username\(3\)**](#) — Determine state of a specific session
[**sd_session_get_vt\(3\)**](#) — Determine state of a specific session
[**sd_session_is_active\(3\)**](#) — Determine state of a specific session
[**sd_session_is_remote\(3\)**](#) — Determine state of a specific session
[**sd_uid_get_display\(3\)**](#) — Determine login state of a specific Unix user ID
[**sd_uid_get_login_time\(3\)**](#) — Determine login state of a specific Unix user ID
[**sd_uid_get_seats\(3\)**](#) — Determine login state of a specific Unix user ID
[**sd_uid_get_sessions\(3\)**](#) — Determine login state of a specific Unix user ID
[**sd_uid_get_state\(3\)**](#) — Determine login state of a specific Unix user ID
[**sd_uid_is_on_seat\(3\)**](#) — Determine login state of a specific Unix user ID
[**SD_WARNING\(3\)**](#) — APIs for new-style daemons
[**sd_watchdog_enabled\(3\)**](#) — Check whether the service manager expects watchdog keep-alive notifications from a service
[**shutdown\(8\)**](#) — Halt, power off or reboot the machine
[**sleep.conf.d\(5\)**](#) — Suspend and hibernation configuration file
[**smbios-type-11\(7\)**](#) — SMBIOS Type 11 strings
[**sysctl.d\(5\)**](#) — Configure kernel parameters at boot
[**system.conf.d\(5\)**](#) — System and session service manager configuration files
[**systemctl\(1\)**](#) — Control the systemd system and service manager
[**systemd\(1\)**](#) — systemd system and service manager
[**systemd-ac-power\(1\)**](#) — Report whether we are connected to an external power source
[**systemd-analyze\(1\)**](#) — Analyze and debug system manager
[**systemd-ask-password\(1\)**](#) — Query the user for a system password
[**systemd-ask-password-console.path\(8\)**](#) — Query the user for system passwords on the console and via wall
[**systemd-ask-password-console.service\(8\)**](#) — Query the user for system passwords on the console and via wall
[**systemd-ask-password-wall.path\(8\)**](#) — Query the user for system passwords on the console and via wall
[**systemd-ask-password-wall.service\(8\)**](#) — Query the user for system passwords on the console and via wall
[**systemd-backlight\(8\)**](#) — Load and save the display backlight brightness at boot and shutdown
[**systemd-backlight@.service\(8\)**](#) — Load and save the display backlight brightness at boot and shutdown
[**systemd-battery-check\(8\)**](#) — Check battery level whether there's enough charge, and power off if not
[**systemd-battery-check.service\(8\)**](#) — Check battery level whether there's enough



charge, and power off if not

[systemd-binfmt\(8\)](#) — Configure additional binary formats for executables at boot

[systemd-binfmt.service\(8\)](#) — Configure additional binary formats for executables at boot

[systemd-bless-boot\(8\)](#) — Mark current boot process as successful

[systemd-bless-boot-generator\(8\)](#) — Pull systemd-bless-boot.service into the initial boot transaction when boot counting is in effect

[systemd-bless-boot.service\(8\)](#) — Mark current boot process as successful

[systemd-boot\(7\)](#) — A simple UEFI boot manager

[systemd-boot-check-no-failures\(8\)](#) — verify that the system booted up cleanly

[systemd-boot-check-no-failures.service\(8\)](#) — verify that the system booted up cleanly

[systemd-boot-random-seed.service\(8\)](#) — Refresh boot loader random seed at boot

[systemd-bsod\(8\)](#) — Displays boot-time emergency log message in full screen.

[systemd-bsod.service\(8\)](#) — Displays boot-time emergency log message in full screen.

[systemd-cat\(1\)](#) — Connect a pipeline or program's output with the journal

[systemd-cgls\(1\)](#) — Recursively show control group contents

[systemd-cgtop\(1\)](#) — Show top control groups by their resource usage

[systemd-confext\(8\)](#) — Activates System Extension Images

[systemd-confext.service\(8\)](#) — Activates System Extension Images

[systemd-coredump\(8\)](#) — Acquire, save and process core dumps

[systemd-coredump.socket\(8\)](#) — Acquire, save and process core dumps

[systemd-coredump@.service\(8\)](#) — Acquire, save and process core dumps

[systemd-creds\(1\)](#) — Lists, shows, encrypts and decrypts service credentials

[systemd-cryptenroll\(1\)](#) — Enroll PKCS#11, FIDO2, TPM2 token/devices to LUKS2 encrypted volumes

[systemd-cryptsetup\(8\)](#) — Full disk decryption logic

[systemd-cryptsetup-generator\(8\)](#) — Unit generator for /etc/crypttab

[systemd-cryptsetup@.service\(8\)](#) — Full disk decryption logic

[systemd-debug-generator\(8\)](#) — Generator for enabling a runtime debug shell and masking specific units at boot

[systemd-delta\(1\)](#) — Find overridden configuration files

[systemd-detect-virt\(1\)](#) — Detect execution in a virtualized environment

[systemd-dissect\(1\)](#) — Dissect Discoverable Disk Images (DDIs)

[systemd-environment-d-generator\(8\)](#) — Load variables specified by environment.d

[systemd-escape\(1\)](#) — Escape strings for usage in systemd unit names

[systemd-firstboot\(1\)](#) — Initialize basic system settings on or before the first boot-up of a system

[systemd-firstboot.service\(1\)](#) — Initialize basic system settings on or before the first boot-up of a system

[systemd-fsck\(8\)](#) — File system checker logic

[systemd-fsck-root.service\(8\)](#) — File system checker logic

[systemd-fsck-usr.service\(8\)](#) — File system checker logic

[systemd-fsck@.service\(8\)](#) — File system checker logic

[systemd-fstab-generator\(8\)](#) — Unit generator for /etc/fstab

[systemd-getty-generator\(8\)](#) — Generator for enabling getty instances on the console

[systemd-gpt-auto-generator\(8\)](#) — Generator for automatically discovering and mounting root, /home/ , /srv/ , /var/ and /var/tmp/ partitions, as well as discovering and enabling swap partitions, based on GPT partition type GUIDs

[systemd-growfs\(8\)](#) — Creating and growing file systems on demand



[**systemd-growfs-root.service\(8\)**](#) — Creating and growing file systems on demand
[**systemd-growfs@.service\(8\)**](#) — Creating and growing file systems on demand
[**systemd-halt.service\(8\)**](#) — System shutdown logic
[**systemd-hibernate-resume\(8\)**](#) — Resume from hibernation
[**systemd-hibernate-resume-generator\(8\)**](#) — Unit generator for resume= kernel parameter
[**systemd-hibernate-resume.service\(8\)**](#) — Resume from hibernation
[**systemd-hibernate.service\(8\)**](#) — System sleep state logic
[**systemd-homed\(8\)**](#) — Home Area/User Account Manager
[**systemd-homed.service\(8\)**](#) — Home Area/User Account Manager
[**systemd-hostnamed\(8\)**](#) — Daemon to control system hostname from programs
[**systemd-hostnamed.service\(8\)**](#) — Daemon to control system hostname from programs
[**systemd-hwdb\(8\)**](#) — hardware database management tool
[**systemd-hybrid-sleep.service\(8\)**](#) — System sleep state logic
[**systemd-id128\(1\)**](#) — Generate and print sd-128 identifiers
[**systemd-importd\(8\)**](#) — VM and container image import and export service
[**systemd-importd.service\(8\)**](#) — VM and container image import and export service
[**systemd-inhibit\(1\)**](#) — Execute a program with an inhibition lock taken
[**systemd-initctl\(8\)**](#) — /dev/initctl compatibility
[**systemd-initctl.service\(8\)**](#) — /dev/initctl compatibility
[**systemd-initctl.socket\(8\)**](#) — /dev/initctl compatibility
[**systemd-integritysetup\(8\)**](#) — Disk integrity protection logic
[**systemd-integritysetup-generator\(8\)**](#) — Unit generator for integrity protected block devices
[**systemd-integritysetup@.service\(8\)**](#) — Disk integrity protection logic
[**systemd-journal-gatewayd\(8\)**](#) — HTTP server for journal events
[**systemd-journal-gatewayd.service\(8\)**](#) — HTTP server for journal events
[**systemd-journal-gatewayd.socket\(8\)**](#) — HTTP server for journal events
[**systemd-journal-remote\(8\)**](#) — Receive journal messages over the network
[**systemd-journal-remote.service\(8\)**](#) — Receive journal messages over the network
[**systemd-journal-remote.socket\(8\)**](#) — Receive journal messages over the network
[**systemd-journal-upload\(8\)**](#) — Send journal messages over the network
[**systemd-journal-upload.service\(8\)**](#) — Send journal messages over the network
[**systemd-journald\(8\)**](#) — Journal service
[**systemd-journald-audit.socket\(8\)**](#) — Journal service
[**systemd-journald-dev-log.socket\(8\)**](#) — Journal service
[**systemd-journald-varlink@.socket\(8\)**](#) — Journal service
[**systemd-journald.service\(8\)**](#) — Journal service
[**systemd-journald.socket\(8\)**](#) — Journal service
[**systemd-journald@.service\(8\)**](#) — Journal service
[**systemd-journald@.socket\(8\)**](#) — Journal service
[**systemd-kexec.service\(8\)**](#) — System shutdown logic
[**systemd-located\(8\)**](#) — Locale bus mechanism
[**systemd-located.service\(8\)**](#) — Locale bus mechanism
[**systemd-logind\(8\)**](#) — Login manager
[**systemd-logind.service\(8\)**](#) — Login manager
[**systemd-machine-id-commit.service\(8\)**](#) — Commit a transient machine ID to disk
[**systemd-machine-id-setup\(1\)**](#) — Initialize the machine ID in /etc/machine-id
[**systemd-machined\(8\)**](#) — Virtual machine and container registration manager



[**systemd-machined.service\(8\)**](#) — Virtual machine and container registration manager
[**systemd-makefs\(8\)**](#) — Creating and growing file systems on demand
[**systemd-makefs@.service\(8\)**](#) — Creating and growing file systems on demand
[**systemd-measure\(1\)**](#) — Pre-calculate and sign expected TPM2 PCR values for booted unified kernel images
[**systemd-mkswap@.service\(8\)**](#) — Creating and growing file systems on demand
[**systemd-modules-load\(8\)**](#) — Load kernel modules at boot
[**systemd-modules-load.service\(8\)**](#) — Load kernel modules at boot
[**systemd-mount\(1\)**](#) — Establish and destroy transient mount or auto-mount points
[**systemd-network-generator\(8\)**](#) — Generate network configuration from the kernel command line
[**systemd-network-generator.service\(8\)**](#) — Generate network configuration from the kernel command line
[**systemd-networkd\(8\)**](#) — Network manager
[**systemd-networkd-wait-online\(8\)**](#) — Wait for network to come online
[**systemd-networkd-wait-online.service\(8\)**](#) — Wait for network to come online
[**systemd-networkd-wait-online@.service\(8\)**](#) — Wait for network to come online
[**systemd-networkd.service\(8\)**](#) — Network manager
[**systemd-notify\(1\)**](#) — Notify service manager about start-up completion and other daemon status changes
[**systemd-nspawn\(1\)**](#) — Spawn a command or OS in a light-weight container
[**systemd-oomd\(8\)**](#) — A userspace out-of-memory (OOM) killer
[**systemd-oomd.service\(8\)**](#) — A userspace out-of-memory (OOM) killer
[**systemd-path\(1\)**](#) — List and query system and user paths
[**systemd-pcrextend\(8\)**](#) — Measure boot phase into TPM2 PCR 11, machine ID and file system identity into PCR 15
[**systemd-pcrfs-root.service\(8\)**](#) — Measure boot phase into TPM2 PCR 11, machine ID and file system identity into PCR 15
[**systemd-pcrfs@.service\(8\)**](#) — Measure boot phase into TPM2 PCR 11, machine ID and file system identity into PCR 15
[**systemd-pcrlock\(8\)**](#) — Analyze and predict TPM2 PCR states and generate an access policy from the prediction
[**systemd-pcrlock-file-system.service\(8\)**](#) — Analyze and predict TPM2 PCR states and generate an access policy from the prediction
[**systemd-pcrlock-firmware-code.service\(8\)**](#) — Analyze and predict TPM2 PCR states and generate an access policy from the prediction
[**systemd-pcrlock-firmware-config.service\(8\)**](#) — Analyze and predict TPM2 PCR states and generate an access policy from the prediction
[**systemd-pcrlock-machine-id.service\(8\)**](#) — Analyze and predict TPM2 PCR states and generate an access policy from the prediction
[**systemd-pcrlock-make-policy.service\(8\)**](#) — Analyze and predict TPM2 PCR states and generate an access policy from the prediction

[**systemd-pcrlock-secureboot-authority.service\(8\)**](#) — Analyze and predict TPM2 PCR states and generate an access policy from the prediction
[**systemd-pcrlock-secureboot-policy.service\(8\)**](#) — Analyze and predict TPM2 PCR states and generate an access policy from the prediction
[**systemd-pcramachine.service\(8\)**](#) — Measure boot phase into TPM2 PCR 11, machine ID and file system identity into PCR 15
[**systemd-pcrphase-initrd.service\(8\)**](#) — Measure boot phase into TPM2 PCR 11,

machine ID and file system identity into PCR 15

[systemd-pcrphase-sysinit.service\(8\)](#) — Measure boot phase into TPM2 PCR 11,

machine ID and file system identity into PCR 15

[systemd-pcrphase.service\(8\)](#) — Measure boot phase into TPM2 PCR 11, machine ID and file system identity into PCR 15

[systemd-portable\(8\)](#) — Portable service manager

[systemd-portable.service\(8\)](#) — Portable service manager

[systemd-poweroff.service\(8\)](#) — System shutdown logic

[systemd-pstore\(8\)](#) — A service to archive contents of pstore

[systemd-pstore.service\(8\)](#) — A service to archive contents of pstore

[systemd-quotacheck\(8\)](#) — File system quota checker logic

[systemd-quotacheck.service\(8\)](#) — File system quota checker logic

[systemd-random-seed\(8\)](#) — Load and save the OS system random seed at boot and shutdown

[systemd-random-seed.service\(8\)](#) — Load and save the OS system random seed at boot and shutdown

[systemd-rc-local-generator\(8\)](#) — Compatibility generator and service to start /etc/rc.local during boot

[systemd-reboot.service\(8\)](#) — System shutdown logic

[systemd-remount-fs\(8\)](#) — Remount root and kernel file systems

[systemd-remount-fs.service\(8\)](#) — Remount root and kernel file systems

[systemd-repart\(8\)](#) — Automatically grow and add partitions

[systemd-repart.service\(8\)](#) — Automatically grow and add partitions

[systemd-resolved\(8\)](#) — Network Name Resolution manager

[systemd-resolved.service\(8\)](#) — Network Name Resolution manager

[systemd-rfkill\(8\)](#) — Load and save the RF kill switch state at boot and change

[systemd-rfkill.service\(8\)](#) — Load and save the RF kill switch state at boot and change

[systemd-rfkill.socket\(8\)](#) — Load and save the RF kill switch state at boot and change

[systemd-run\(1\)](#) — Run programs in transient scope units, service units, or path-, socket-, or timer-triggered service units

[systemd-run-generator\(8\)](#) — Generator for invoking commands specified on the kernel command line as system service

[systemd-shutdown\(8\)](#) — System shutdown logic

[systemd-sleep\(8\)](#) — System sleep state logic

[systemd-sleep.conf\(5\)](#) — Suspend and hibernation configuration file

[systemd-socket-activate\(1\)](#) — Test socket activation of daemons

[systemd-socket-proxd\(8\)](#) — Bidirectionally proxy local sockets to another (possibly remote) socket

[systemd-soft-reboot.service\(8\)](#) — Userspace reboot operation

[systemd-stdio-bridge\(1\)](#) — D-Bus proxy

[systemd-storagetm\(8\)](#) — Exposes all local block devices as NVMe-TCP mass storage devices

[systemd-storagetm.service\(8\)](#) — Exposes all local block devices as NVMe-TCP mass storage devices

[systemd-stub\(7\)](#) — A simple UEFI kernel boot stub

[systemd-suspend-then-hibernate.service\(8\)](#) — System sleep state logic

[systemd-suspend.service\(8\)](#) — System sleep state logic

[systemd-sysctl\(8\)](#) — Configure kernel parameters at boot

[systemd-sysctl.service\(8\)](#) — Configure kernel parameters at boot



[**systemd-sysext\(8\)**](#) — Activates System Extension Images
[**systemd-sysext.service\(8\)**](#) — Activates System Extension Images
[**systemd-system-update-generator\(8\)**](#) — Generator for redirecting boot to offline update mode
[**systemd-system.conf\(5\)**](#) — System and session service manager configuration files
[**systemd-sysupdate\(8\)**](#) — Automatically Update OS or Other Resources
[**systemd-sysupdate-reboot.service\(8\)**](#) — Automatically Update OS or Other Resources
[**systemd-sysupdate-reboot.timer\(8\)**](#) — Automatically Update OS or Other Resources
[**systemd-sysupdate.service\(8\)**](#) — Automatically Update OS or Other Resources
[**systemd-sysupdate.timer\(8\)**](#) — Automatically Update OS or Other Resources
[**systemd-sysusers\(8\)**](#) — Allocate system users and groups
[**systemd-sysusers.service\(8\)**](#) — Allocate system users and groups
[**systemd-sysv-generator\(8\)**](#) — Unit generator for SysV init scripts
[**systemd-time-wait-sync\(8\)**](#) — Wait until kernel time is synchronized
[**systemd-time-wait-sync.service\(8\)**](#) — Wait until kernel time is synchronized
[**systemd-timedated\(8\)**](#) — Time and date bus mechanism
[**systemd-timedated.service\(8\)**](#) — Time and date bus mechanism
[**systemd-timesyncd\(8\)**](#) — Network Time Synchronization
[**systemd-timesyncd.service\(8\)**](#) — Network Time Synchronization
[**systemd-tmpfiles\(8\)**](#) — Creates, deletes and cleans up volatile and temporary files and directories
[**systemd-tmpfiles-clean.service\(8\)**](#) — Creates, deletes and cleans up volatile and temporary files and directories
[**systemd-tmpfiles-clean.timer\(8\)**](#) — Creates, deletes and cleans up volatile and temporary files and directories
[**systemd-tmpfiles-setup-dev-early.service\(8\)**](#) — Creates, deletes and cleans up volatile and temporary files and directories
[**systemd-tmpfiles-setup-dev.service\(8\)**](#) — Creates, deletes and cleans up volatile and temporary files and directories
[**systemd-tmpfiles-setup.service\(8\)**](#) — Creates, deletes and cleans up volatile and temporary files and directories
[**systemd-tpm2-setup\(8\)**](#) — Set up the TPM2 Storage Root Key (SRK) at boot
[**systemd-tpm2-setup-early.service\(8\)**](#) — Set up the TPM2 Storage Root Key (SRK) at boot
[**systemd-tpm2-setup.service\(8\)**](#) — Set up the TPM2 Storage Root Key (SRK) at boot
[**systemd-tty-ask-password-agent\(1\)**](#) — List or process pending systemd password requests
[**systemd-udev-settle.service\(8\)**](#) — Wait for all pending udev events to be handled
[**systemd-udevd\(8\)**](#) — Device event managing daemon
[**systemd-udevd-control.socket\(8\)**](#) — Device event managing daemon
[**systemd-udevd-kernel.socket\(8\)**](#) — Device event managing daemon
[**systemd-udevd.service\(8\)**](#) — Device event managing daemon
[**systemd-umount\(1\)**](#) — Establish and destroy transient mount or auto-mount points
[**systemd-update-done\(8\)**](#) — Mark /etc/ and /var/ fully updated
[**systemd-update-done.service\(8\)**](#) — Mark /etc/ and /var/ fully updated
[**systemd-update-utmp\(8\)**](#) — Write audit and utmp updates at bootup, runlevel changes and shutdown
[**systemd-update-utmp-runlevel.service\(8\)**](#) — Write audit and utmp updates at bootup, runlevel changes and shutdown



[**systemd-update-utmp.service\(8\)**](#) — Write audit and utmp updates at bootup, runlevel changes and shutdown
[**systemd-user-runtime-dir\(5\)**](#) — System units to start the user manager
[**systemd-user-sessions\(8\)**](#) — Permit user logins after boot, prohibit user logins at shutdown
[**systemd-user-sessions.service\(8\)**](#) — Permit user logins after boot, prohibit user logins at shutdown
[**systemd-user.conf\(5\)**](#) — System and session service manager configuration files
[**systemd-userdbd\(8\)**](#) — JSON User/Group Record Query Multiplexer/NSS
Compatibility
[**systemd-userdbd.service\(8\)**](#) — JSON User/Group Record Query Multiplexer/NSS
Compatibility
[**systemd-vconsole-setup\(8\)**](#) — Configure the virtual consoles
[**systemd-vconsole-setup.service\(8\)**](#) — Configure the virtual consoles
[**systemd-veritysetup\(8\)**](#) — Disk verity protection logic
[**systemd-veritysetup-generator\(8\)**](#) — Unit generator for verity protected block devices
[**systemd-veritysetup@.service\(8\)**](#) — Disk verity protection logic
[**systemd-vmspawn\(1\)**](#) — Spawn an OS in a virtual machine.
[**systemd-volatile-root\(8\)**](#) — Make the root file system volatile
[**systemd-volatile-root.service\(8\)**](#) — Make the root file system volatile
[**systemd-xdg-autostart-generator\(8\)**](#) — User unit generator for XDG autostart files
[**systemd.automount\(5\)**](#) — Automount unit configuration
[**systemd.device\(5\)**](#) — Device unit configuration
[**systemd.directives\(7\)**](#) — Index of configuration directives
[**systemd.dnssd\(5\)**](#) — DNS-SD configuration
[**systemd.environment-generator\(7\)**](#) — systemd environment file generators
[**systemd.exec\(5\)**](#) — Execution environment configuration
[**systemd.generator\(7\)**](#) — systemd unit generators
[**systemd.image-policy\(7\)**](#) — Disk Image Dissection Policy
[**systemd.journal-fields\(7\)**](#) — Special journal fields
[**systemd.kill\(5\)**](#) — Process killing procedure configuration
[**systemd.link\(5\)**](#) — Network device configuration
[**systemd.mount\(5\)**](#) — Mount unit configuration
[**systemd.negative\(5\)**](#) — DNSSEC trust anchor configuration files
[**systemd.net-naming-scheme\(7\)**](#) — Network device naming schemes
[**systemd.netdev\(5\)**](#) — Virtual Network Device configuration
[**systemd.network\(5\)**](#) — Network configuration
[**systemd.nspawn\(5\)**](#) — Container settings
[**systemd.offline-updates\(7\)**](#) — Implementation of offline updates in systemd
[**systemd.path\(5\)**](#) — Path unit configuration
[**systemd.pcrlock\(5\)**](#) — PCR measurement prediction files
[**systemd.pcrlock.d\(5\)**](#) — PCR measurement prediction files
[**systemd.positive\(5\)**](#) — DNSSEC trust anchor configuration files
[**systemd.preset\(5\)**](#) — Service enablement presets
[**systemd.resource-control\(5\)**](#) — Resource control unit settings
[**systemd.scope\(5\)**](#) — Scope unit configuration
[**systemd.service\(5\)**](#) — Service unit configuration
[**systemd.slice\(5\)**](#) — Slice unit configuration
[**systemd.socket\(5\)**](#) — Socket unit configuration



[**systemd.special\(7\)**](#) — Special systemd units
[**systemd.swap\(5\)**](#) — Swap unit configuration
[**systemd.syntax\(7\)**](#) — General syntax of systemd configuration files
[**systemd.system-credentials\(7\)**](#) — System Credentials
[**systemd.target\(5\)**](#) — Target unit configuration
[**systemd.time\(7\)**](#) — Time and date specifications
[**systemd.timer\(5\)**](#) — Timer unit configuration
[**systemd.unit\(5\)**](#) — Unit configuration
[**sysupdate.d\(5\)**](#) — Transfer Definition Files for Automatic Updates
[**sysusers.d\(5\)**](#) — Declarative allocation of system users and groups

T

[**telinit\(8\)**](#) — Change SysV runlevel
[**timedatectl\(1\)**](#) — Control the system time and date
[**timesyncd.conf\(5\)**](#) — Network Time Synchronization configuration files
[**timesyncd.conf.d\(5\)**](#) — Network Time Synchronization configuration files
[**tmpfiles.d\(5\)**](#) — Configuration for creation, deletion and cleaning of volatile and temporary files

U

[**udev\(7\)**](#) — Dynamic device management
[**udev.conf\(5\)**](#) — Configuration for device event managing daemon
[**udev_device_get_action\(3\)**](#) — Query device properties
[**udev_device_get_current_tags_list_entry\(3\)**](#) — Retrieve or set device attributes
[**udev_device_get_devlinks_list_entry\(3\)**](#) — Retrieve or set device attributes
[**udev_device_get_devnode\(3\)**](#) — Query device properties
[**udev_device_get_devnum\(3\)**](#) — Query device properties
[**udev_device_get_devpath\(3\)**](#) — Query device properties
[**udev_device_get_devtype\(3\)**](#) — Query device properties
[**udev_device_get_driver\(3\)**](#) — Query device properties
[**udev_device_get_is_initialized\(3\)**](#) — Query device properties
[**udev_device_get_parent\(3\)**](#) — Query device properties
[**udev_device_get_parent_with_subsystem_devtype\(3\)**](#) — Query device properties
[**udev_device_get_properties_list_entry\(3\)**](#) — Retrieve or set device attributes
[**udev_device_get_property_value\(3\)**](#) — Retrieve or set device attributes
[**udev_device_get_subsystem\(3\)**](#) — Query device properties
[**udev_device_get_sysattr_list_entry\(3\)**](#) — Retrieve or set device attributes
[**udev_device_get_sysattr_value\(3\)**](#) — Retrieve or set device attributes
[**udev_device_get_sysname\(3\)**](#) — Query device properties
[**udev_device_get_sysnum\(3\)**](#) — Query device properties
[**udev_device_get_syspath\(3\)**](#) — Query device properties
[**udev_device_get_tags_list_entry\(3\)**](#) — Retrieve or set device attributes
[**udev_device_get_udev\(3\)**](#) — Query device properties
[**udev_device_has_current_tag\(3\)**](#) — Retrieve or set device attributes
[**udev_device_has_tag\(3\)**](#) — Retrieve or set device attributes
[**udev_device_new_from_device_id\(3\)**](#) — Create, acquire and release a udev device



object

[udev_device_new_from_devnum\(3\)](#) — Create, acquire and release a udev device object

[udev_device_new_from_environment\(3\)](#) — Create, acquire and release a udev device object

[udev_device_new_from_subsystem_sysname\(3\)](#) — Create, acquire and release a udev device object

[udev_device_new_from_syspath\(3\)](#) — Create, acquire and release a udev device object

[udev_device_ref\(3\)](#) — Create, acquire and release a udev device object

[udev_device_set_sysattr_value\(3\)](#) — Retrieve or set device attributes

[udev_device_unref\(3\)](#) — Create, acquire and release a udev device object

[udev_enumerate_add_match_is_initialized\(3\)](#) — Modify filters

[udev_enumerate_add_match_parent\(3\)](#) — Modify filters

[udev_enumerate_add_match_property\(3\)](#) — Modify filters

[udev_enumerate_add_match_subsystem\(3\)](#) — Modify filters

[udev_enumerate_add_match_sysattr\(3\)](#) — Modify filters

[udev_enumerate_add_match_sysname\(3\)](#) — Modify filters

[udev_enumerate_add_match_tag\(3\)](#) — Modify filters

[udev_enumerate_add_nomatch_subsystem\(3\)](#) — Modify filters

[udev_enumerate_add_nomatch_sysattr\(3\)](#) — Modify filters

[udev_enumerate_add_syspath\(3\)](#) — Query or modify a udev enumerate object

[udev_enumerate_get_list_entry\(3\)](#) — Query or modify a udev enumerate object

[udev_enumerate_get_udev\(3\)](#) — Query or modify a udev enumerate object

[udev_enumerate_new\(3\)](#) — Create, acquire and release a udev enumerate object

[udev_enumerate_ref\(3\)](#) — Create, acquire and release a udev enumerate object

[udev_enumerate_scan_devices\(3\)](#) — Query or modify a udev enumerate object

[udev_enumerate_scan_subsystems\(3\)](#) — Query or modify a udev enumerate object

[udev_enumerate_unref\(3\)](#) — Create, acquire and release a udev enumerate object

[udev_list_entry\(3\)](#) — Iterate and access udev lists

[udev_list_entry_get_by_name\(3\)](#) — Iterate and access udev lists

[udev_list_entry_get_name\(3\)](#) — Iterate and access udev lists

[udev_list_entry_get_next\(3\)](#) — Iterate and access udev lists

[udev_list_entry_get_value\(3\)](#) — Iterate and access udev lists

[udev_monitor_enable_receiving\(3\)](#) — Query and modify device monitor

[udev_monitor_filter_add_match_subsystem_devtype\(3\)](#) — Modify filters

[udev_monitor_filter_add_match_tag\(3\)](#) — Modify filters

[udev_monitor_filter_remove\(3\)](#) — Modify filters

[udev_monitor_filter_update\(3\)](#) — Modify filters

[udev_monitor_get_fd\(3\)](#) — Query and modify device monitor

[udev_monitor_get_udev\(3\)](#) — Query and modify device monitor

[udev_monitor_new_from_netlink\(3\)](#) — Create, acquire and release a udev monitor object

[udev_monitor_receive_device\(3\)](#) — Query and modify device monitor

[udev_monitor_ref\(3\)](#) — Create, acquire and release a udev monitor object

[udev_monitor_set_receive_buffer_size\(3\)](#) — Query and modify device monitor

[udev_monitor_unref\(3\)](#) — Create, acquire and release a udev monitor object

[udev_new\(3\)](#) — Create, acquire and release a udev context object

[udev_ref\(3\)](#) — Create, acquire and release a udev context object

[udev_unref\(3\)](#) — Create, acquire and release a udev context object



[udevadm\(8\)](#) — udev management tool

[ukify\(1\)](#) — Combine components into a signed Unified Kernel Image for UEFI systems

[user-runtime-dir@.service\(5\)](#) — System units to start the user manager

[user.conf.d\(5\)](#) — System and session service manager configuration files

[user@.service\(5\)](#) — System units to start the user manager

[userdbctl\(1\)](#) — Inspect users, groups and group memberships

V

[varlinkctl\(1\)](#) — Introspect with and invoke Varlink services

[vconsole.conf\(5\)](#) — Configuration file for the virtual console

[veritytab\(5\)](#) — Configuration for verity block devices

See Also

[systemd.directives\(7\)](#)

This index contains 1157 entries, referring to 397 individual manual pages.



* Table of Contents :TOC: - [[#intro][Intro]] - [[#getting-started][Getting started]] - [[#telling-zsh-which-function-to-use-for-completing-a-command][Telling zsh which function to use for completing a command]] - [[#completing-generic-gnu-commands][Completing generic gnu commands]] - [[#copying-completions-from-another-command][Copying completions from another command]] - [[#writing-your-own-completion-functions][Writing your own completion functions]] - [[#utility-functions][Utility functions]] - [[#writing-simple-completion-functions-using-_describe][Writing simple completion functions using _describe]] - [[#writing-completion-functions-using-_alternative][Writing completion functions using _alternative]] - [[#writing-completion-functions-using-_arguments][Writing completion functions using _arguments]] - [[#writing-completion-functions-using-_regex_arguments-and-_regex_words][Writing completion functions using _regex_arguments and _regex_words]] - [[#complex-completions-with-_values-_sep_parts--_multi_parts][complex completions with _values, _sep_parts, & _multi_parts]] - [[#adding-completion-words-directly-using-compadd][Adding completion words directly using compadd]] - [[#testing--debugging][Testing & debugging]] - [[#gotchas-things-to-watch-out-for][Gotchas (things to watch out for)]] - [[#tips][Tips]] - [[#other-resources][Other resources]] * Intro The official documentation for writing zsh completion functions is difficult to understand, and doesn't give many examples. At the time of writing this document I was able to find a few other tutorials on the web, however those tutorials only explain a small portion of the capabilities of the completion system. This document aims to cover areas not explained elsewhere, with examples, so that you can learn how to write more advanced completion functions. I do not go into all the details, but will give enough information and examples to get you up and running. If you need more details you can look it up for yourself in the [[<https://zsh.sourceforge.net/Doc/Release/Completion-System.html#Completion-System>][official documentation]]. Please make any scripts that you create publicly available for others (e.g. by forking this repo and making a [[id:64bcd501-b0f0-48c7-b8e2-07af708b95ec][pull request]]). Also if you have any more information to add or improvements to make to this tutorial, please do. * Getting started ** Telling zsh which function to use for completing a command Completion functions for commands are stored in files with names beginning with an underscore _, and these files should be placed in a directory listed in the \$fpath variable. You can add a directory to \$fpath by adding a line like this to your ~/.zshrc file:
#+BEGIN_SRC sh fpath=(~/newdir \$fpath) #+END_SRC The first line of a completion function file can look something like this: #+BEGIN_SRC sh #compdef foobar #+END_SRC This tells zsh that the file contains code for completing the foobar command. This is the format that you will use most often for the first line, but you can also use the same file for completing several different functions if you want. See [[<https://zsh.sourceforge.net/Doc/Release/Completion-System.html#Autoloaded-files>][here]] for more details. You can also use the compdef command directly (e.g. in your ~/.zshrc file) to tell zsh which function to use for completing a command like this: #+BEGIN_SRC sh > compdef _function foobar #+END_SRC or to use the same completions for several commands: #+BEGIN_SRC sh > compdef _function foobar goocar hoodar #+END_SRC or if you want to supply arguments: #+BEGIN_SRC sh > compdef '_function arg1 arg2' foobar #+END_SRC See [[<https://zsh.sourceforge.net/Doc/Release/Completion-System.html#Functions-4>][here]] for more details. ** Completing generic gnu commands Many [[<https://www.gnu.org/>][gnu]] commands have a standardized way of listing option descriptions (when the --help option is used). For these commands you can use the _gnu_generic function for automatically creating completions, like this: #+BEGIN_SRC sh > compdef _gnu_generic foobar #+END_SRC or to use _gnu_generic with several different commands: #+BEGIN_SRC sh > compdef _gnu_generic foobar goocar hoodar #+END_SRC This line can be placed in your ~/.zshrc file. ** Copying completions from another command If you want a command, say cmd1, to have the same completions as another, say cmd2, which has already had completions defined for it, you can do this: #+BEGIN_SRC sh > compdef cmd1=cmd2 #+END_SRC This can be useful for example if you have created an alias for a command to help you remember it. * Writing your own completion functions A good way to get started is to look at some already defined completion functions. On my linux installation these are found in /usr/share/zsh/functions/Completion/Unix and /usr/share/zsh/functions/Completion/Linux and a few other subdirs. You will notice that the _arguments function is used a lot in these files. This is a utility function that makes it easy to write simple completion functions. The _arguments function is a wrapper around the compadd builtin function. The compadd builtin is the core function used to add completion words to the command line, and control its behaviour. However, most of the time you will not need to use compadd, since there are many utility functions such as _arguments and _describe which are easier to use. For very basic completions the _describe function should be adequate ** Utility functions Here is a list of some of the utility functions that may be of use. The full list of utility functions, with full explanations, is available [[<https://zsh.sourceforge.net/Doc/Release/Completion-System.html#Completion-Functions>][here]]. Examples of how to use these functions are given in the next section. *** main utility functions for overall completion | _alternative | Can be used to generate completion candidates from other utility functions or shell code. || _arguments | Used to specify how to complete individual options & arguments for a command with unix style options. || _describe | Used for creating simple completions consisting of words with descriptions (but no actions). Easier to use than _arguments || _gnu_generic | Can be used to complete options for commands that understand the '--help' option. || _regex_arguments | Creates a function for matching commandline arguments with regular expressions, and then performing actions/completions. | *** functions for performing complex completions of single words | _values | Used for completing arbitrary keywords (values) and their arguments, or comma separated lists of such combinations. || _combination | Used to complete combinations of values, for example pairs of hostnames and usernames. || _multi_parts | Used for completing multiple parts of words separately where each part is

separated by some char, e.g. for completing partial filepaths: /u/i/sy -> /usr/include/sys || _sep_parts | Like _multi_parts but allows different separators at different parts of the completion. || _sequence | Used as a wrapper around another completion function to complete a delimited list of matches generated by that other function. *** functions for completing specific types of objects | _path_files | Used to complete filepaths. Take several options to control behaviour. || _files | Calls _path_files with all options except -g and -. These options depend on file-patterns style setting. || _net_interfaces | Used for completing network interface names || _users | Used for completing user names || _groups | Used for completing group names || _options | Used for completing the names of shell options. || _parameters | Used for completing the names of shell parameters/variables (can restrict to those matching a pattern). | *** functions for handling cached completions If you have a very large number of completions you can save them in a cache file so that the completions load quickly. | _cache_invalid | indicates whether the completions cache corresponding to a given cache identifier needs rebuilding || _retrieve_cache | retrieves completion information from a cache file || _store_cache | store completions corresponding to a given cache identifier in a cache file | *** other functions | _message | Used for displaying help messages in places where no completions can be generated. || _regex_words | Can be used to generate arguments for the _regex_arguments command. This is easier than writing the arguments manually. || _guard | Can be used in the ACTION of specifications for _arguments and similar functions to check the word being completed. | *** Actions Many of the utility functions such as _arguments, _regex_arguments, _alternative and _values may include an action at the end of an option/argument specification. This action indicates how to complete the corresponding argument. The actions can take one of the following forms: | () | Argument is required but no matches are generated for it. || (ITEM1 ITEM2) | List of possible matches || ((ITEM1:'DESC1' ITEM2:'DESC2')) | List of possible matches, with descriptions. Make sure to use different quotes than those around the whole specification. || ->STRING | Set \$state to STRING and continue (\$state can be checked in a case statement after the utility function call) || FUNCTION | Name of a function to call for generating matches or performing some other action, e.g. _files or _message || {EVAL-STRING} | Evaluate string as shell code to generate matches. This can be used to call a utility function with arguments, e.g. _values or _describe || =ACTION | Inserts a dummy word into completion command line without changing the point at which completion takes place. | Not all action types are available for all utility functions that use them. For example the ->STRING type is not available in the _regex_arguments or _alternative functions. ** Writing simple completion functions using _describe The _describe function can be used for simple completions where the order and position of the options/arguments is not important. You just need to create an array parameter to hold the options & their descriptions, and then pass the parameter name as an argument to _describe. The following example creates completion candidates c and d, with the descriptions (note this should be put in a file called _cmd in some directory listed in \$fpath).
#+BEGIN_SRC sh #compdef cmd local -a subcmds subcmds=(c:description for c command' 'd:description for d command') _describe 'command' subcmds #+END_SRC You can use several different lists separated by a double hyphen as follows but note that this mixes the matches under and single heading and is not intended to be used with different types of completion candidates: #+BEGIN_SRC sh local -a subcmds topics subcmds=(c:description for c command' 'd:description for d command') topics=(e:description for e help topic' 'f:description for f help topic') _describe 'command' subcmds -- topics #+END_SRC If two candidates have the same description, _describe collects them together on the same row and ensures that descriptions are aligned in neatly in columns. The _describe function can be used in an ACTION as part of a specification for _alternative, _arguments or _regex_arguments. In this case you will have to put it in braces with its arguments, e.g. 'TAG:DESCRIPTION:{ _describe 'values' options}' ** Writing completion functions using _alternative Like _describe, this function performs simple completions where the order and position of options/arguments is not important. However, unlike _describe, instead of fixed matches further functions may be called to generate the completion candidates. Furthermore, _alternative allows a mix of different types of completion candidates to be mixed. As arguments it takes a list of specifications each in the form 'TAG:DESCRIPTION:ACTION' where TAG is a special tag that identifies the type of completion matches, DESCRIPTION is used as a heading to describe the group of completion candidates collectively, and ACTION is one of the action types listed previously (apart from the ->STRING and =ACTION forms). For example:
#+BEGIN_SRC sh _alternative 'arguments:custom arg:(a b c)' 'files:filename:_files' #+END_SRC The first specification adds completion candidates a, b & c, and the second specification calls the _files function for completing filepaths. We could split the specifications over several lines with \ and add descriptions to each of the custom args like this: #+BEGIN_SRC sh _alternative \ 'args:custom arg:((a:"description a" b:"description b" c:"description c"))' \ 'files:filename:_files' #+END_SRC If we want to pass arguments to _files they can simply be included, like this: #+BEGIN_SRC sh _alternative \ 'args:custom arg:((a:"description a" b:"description b" c:"description c"))'\ 'files:filename:_files' -/ #+END_SRC To use parameter expansion to create our list of completions we must use double quotes to quote the specifications, e.g.: #+BEGIN_SRC sh _alternative \ "dirs:user directory:(\$userdirs)" \ "pids:process ID:(\$(\$ps -A o pid=))" #+END_SRC In this case the first specification adds the words stored in the \$userdirs variable, and the second specification evaluates 'ps -A o pid=' to get a list of pids to use as completion candidates. In practice, we would make use of the existing _pids function for this. We can use other utility functions such as _values in the ACTION to perform more complex completions, e.g.: #+BEGIN_SRC sh _alternative \ "directories:user directory:(\$userdirs)" \ 'options:comma-separated opt: _values -s , letter a b c' #+END_SRC this will complete the items in \$userdirs, as well as a comma separated list containing a, b &/or c. Note the use of the initial space before _values. This is needed because _values doesn't understand standard compadd options for

descriptions. As with _describe, the _alternative function can itself be used in an ACTION as part of a specification for _arguments or _regex_arguments. ** Writing completion functions using _arguments With a single call to the _arguments function you can create fairly sophisticated completion functions. It is intended to handle typical commands that take a variety of options along with some normal arguments. Like the _alternative function, _arguments takes a list of specification strings as arguments. These specification strings specify options and any corresponding option arguments (e.g. -f filename), or command arguments. Basic option specifications take the form '-OPT[DESCRIPTION]', e.g. like this: #+BEGIN_SRC sh _arguments '-s[sort output]' '--l[long output]' '-l[long output]' #+END_SRC Arguments for the option can be specified after the option description in this form '-OPT[DESCRIPTION]:MESSAGE:ACTION', where MESSAGE is a message to display and ACTION can be any of the forms mentioned in the ACTIONS section above. For example: #+BEGIN_SRC sh _arguments '-f[input file]:filename:_files' #+END_SRC Command argument specifications take the form 'N:MESSAGE:ACTION' where N indicates that it is the Nth command argument, and MESSAGE & ACTION are as before. If the N is omitted then it just means the next command argument (after any that have already been specified). If a double colon is used at the start (after N) then the argument is optional. For example: #+BEGIN_SRC sh _arguments '-s[sort output]' '1:first arg:_net_interfaces' '::optional arg:_files' ':next arg:(a b c)' #+END_SRC here the first arg is a network interface, the next optional arg is a file name, the last arg can be either a, b or c, and the -s option may be completed at any position. The _arguments function allows the full set of ACTION forms listed in the ACTION section above. This means that you can use actions for selecting case statement branches like this: #+BEGIN_SRC sh _arguments '-m[music file]:filename:->files' '-f[flags]:flag:->flags' case "\$state" in files) local -a music_files music_files=(Music/**/*.{mp3,wav,flac,ogg}) _multi_parts / music_files ;; flags) _values -s , 'flags' a b c d e ;; esac #+END_SRC In this case paths to music files are completed stepwise descending down directories using the _multi_parts function, and the flags are completed as a comma separated list using the _values function. I have just given you the basics of _arguments specifications here, you can also specify mutually exclusive options, repeated options & arguments, options beginning with + instead of -, etc. For more details see the [[<https://zsh.sourceforge.net/Doc/Release/Completion-System.html#Completion-System>][official documentation]]. Also have a look at the tutorials mentioned at the end of this document, and the completion functions in the [[<https://github.com/vapniks/zsh-completions/tree/master/src>][src directory]]. ** Writing completion functions using _regex_arguments and _regex_words If you have a complex command line specification with several different possible argument sequences then the _regex_arguments function may be what you need. It typically works well where you have a series of keywords followed by a variable number of arguments. _regex_arguments creates a completion function whose name is given by the first argument. Hence you need to first call _regex_arguments to create the completion function, and then call that function, e.g. like this: #+BEGIN_SRC sh _regex_arguments _cmd OTHER_ARGS.. _cmd "\$@" #+END_SRC The OTHER_ARGS should be sequences of specifications for matching & completing words on the command line. These sequences can be separated by "}" to represent alternative sequences of words. You can use bracketing to arbitrary depth to specify alternate subsequences, but the brackets must be backslashed like this `(\)` or quoted like this `(' ')'. For example: #+BEGIN_SRC sh _regex_arguments _cmd SEQ1 '| SEQ2 \(`SEQ2a '| SEQ2b`\)` _cmd "\$@" #+END_SRC This specifies a command line matching either SEQ1, or SEQ2 followed by SEQ2a or SEQ2b. You are describing the form arguments to the command take in the form of a regular expression grammar. Each specification in a sequence must contain a / PATTERN/ part at the start followed by an optional ':TAG:DESCRIPTION:ACTION' part. Each PATTERN is a regular expression to match a word on the command line. These patterns are processed sequentially until we reach a pattern that doesn't match at which point any corresponding ACTION is performed to obtain completions for that word. Note that there needs to be a pattern to match the initial command itself. See below for further explanation about PATTERNS. The ':TAG:DESCRIPTION:ACTION' part is interpreted in the same way as for the _alternative function specifications, except that it has an extra : at the start, and now all of the possible ACTION formats listed previously are allowed. Here is an example: #+BEGIN_SRC sh _regex_arguments _cmd /\$'[^\0]##\0' \(`/\$'word1(a|b|c)\0'\)` :word:first word:(word1a word1b word1c)` '|`/\$'word11(a|b|c)\0'\` :word:first word:(word11a word11b word11c)` \|`(\$'word2(a|b|c)\0'\` :word:second word:(word2a word2b word2c)` \|`/\$'word22(a|b|c)\0'\` :word:second word:(word22a word22b word22c)` \|`)_cmd "\$@" #+END_SRC in this case the first word can be word1 or word11 followed by an a, b or c, and if the first word contains 11 then a second word is allowed which can be word2 followed by an a, b, or c, or a filename. If this sounds too complicated a much simpler alternative is to use the _regex_words function for creating specifications for _regex_arguments. *** Patterns You may notice that the / PATTERN/ specs in the previous example don't look like normal regular expressions. Often a string parameter in the form '\$foo\0' is used. This is so that the \0 in the string is interpreted correctly as a null char which is used to separate words in the internal representation. If you don't include the \0 at the end of the pattern you may get problems matching the next word. If you need to use the contents of a variable in a pattern, you can double quote it so that it gets expanded and then put a string parameter containing a null char afterwards, like this: "\$somevar"\\$'\0' The regular expression syntax for patterns seems to be a bit different from normal regular expressions, and I can't find documentation anywhere. However I have managed to work out what the following special chars are for: | * | wildcard - any number of chars || ? | wildcard - single char || # | zero or more of the previous char (like * in a normal regular expression) || ## | one or more of the previous char (like + in a normal regular expression) | *** _regex_words The _regex_words function makes it much easier to create specifications for _regex_arguments. The results of calling

`_regex_words` can be stored in a variable which can then be used instead of a specification for `_regex_arguments`. To create a specification using `_regex_words` you supply it with a tag followed by a description followed by a list of specifications for individual words. These specifications take the form 'WORD:DESCRIPTION:SPEC' where WORD is the word to be completed, DESCRIPTION is a description for it, and SPEC can be another variable created by `_regex_words` specifying words that come after the current word or blank if there are no further words. For example: `#+BEGIN_SRC sh _regex_words firstword 'The first word' 'word1a:a word:' 'word1b:b word:' 'word1c:c word' #+END_SRC` the results of this function call will be stored in the `$reply` array, and so we should store it in another array before `$reply` gets changed again, like this: `#+BEGIN_SRC sh local -a firstword _regex_words word 'The first word' 'word1a:a word:' 'word1b:b word:' 'word1c:c word' firstword="$reply[@]" #+END_SRC` we could then use it with `_regex_arguments` like this: `#+BEGIN_SRC sh _regex_arguments _cmd /$[^0]##0/ "$firstword[@]" _cmd "$@" #+END_SRC` Note that I have added an extra pattern for the initial command word itself. Here is a more complex example where we call `_regex_words` for different words on the command line `#+BEGIN_SRC sh local -a firstword firstword2 secondword secondword2 _regex_words word1 'The second word' 'woo:tang clan' 'hoo:not me' secondword=("$reply[@]") _regex_words word2 'Another second word' 'yee:thou' 'haa:very funny!' secondword2=("$reply[@]") _regex_words commands 'The first word' 'foo:do foo' 'man:yeah man' 'chu:at chu' firstword=("$reply[@]") _regex_words word4 'Another first word' 'boo:scare somebody:$secondword' 'ga:baby noise:$secondword"\ 'loo:go to the toilet:$secondword2 firstword2=("$reply[@]") _regex_arguments _hello /$[^0]##0/ "${firstword[@]}" "${firstword2[@]}" _hello "$@" #+END_SRC` In this case the first word can be one of "foo", "man", "chu", "boo", "ga" or "loo". If the first word is "boo" or "ga" then the second word can be "woo" or "hoo", and if the first word is "loo" then the second word can be "yee" or "haa", in the other cases there is no second word. For a good example of the usage of `_regex_words` have a look at the `_ip` function. ** complex completions with `_values`, `_sep_parts`, & `_multi_parts` The `_values`, `_sep_parts` & `_multi_parts` functions can be used either on their own, or as ACTIONS in specifications for `_alternative`, `_arguments` or `_regex_arguments`. The following examples may be instructive. See the [[<https://zsh.sourceforge.net/Doc/Release/Completion-System.html#Completion-System>]] for more info. Space separated list of mp3 files: `#+BEGIN_SRC sh _values 'mp3 files' ~/*.mp3 #+END_SRC` Comma separated list of session id numbers: `#+BEGIN_SRC sh _values -s , 'session id' "${{(uonzf)}${(ps -A o sid)}}" #+END_SRC` Completes `foo@news:woo`, or `foo@news:laa`, or `bar@news:woo`, etc: `#+BEGIN_SRC sh _sep_parts '(foo bar)' @ '(news ftp)' : '(woo laa)' #+END_SRC` Complete some MAC addresses one octet at a time: `#+BEGIN_SRC sh _multi_parts :` `'(00:11:22:33:44:55 00:23:34:45:56:67 00:23:45:56:67:78)' #+END_SRC` ** Adding completion words directly using `compadd` For more fine grained control you can use the builtin `compadd` function to add completion words directly. This function has many different options for controlling how completions are displayed and how text on the command line can be altered when words are completed. Read the [[<https://zsh.sourceforge.net/Doc/Release/Completion-System.html#Completion-System>]] for full details. Here I just give a few simple examples. Add some words to the list of possible completions: `#+BEGIN_SRC sh compadd foo bar blah #+END_SRC` As above but also display an explanation: `#+BEGIN_SRC sh compadd -X 'Some completions' foo bar blah #+END_SRC` As above but automatically insert a prefix of "what_" before the completed word: `#+BEGIN_SRC sh compadd -P what_ foo bar blah #+END_SRC` As above but automatically insert a suffix of "_todo" after the completed word: `#+BEGIN_SRC sh compadd -S _todo foo bar blah #+END_SRC` As above but automatically remove the "_todo" suffix if a blank char is typed after the suffix: `#+BEGIN_SRC sh compadd -P _todo -q foo bar blah #+END_SRC` Add words in array `$wordsarray` to the list of possible completions `#+BEGIN_SRC sh compadd -a wordsarray #+END_SRC` * Testing & debugging To reload a completion function: `#+BEGIN_SRC sh > unfunction _func > autoload -U _func #+END_SRC` The following functions can be called to obtain useful information. If the default keybindings don't work you can try pressing Alt+x and then enter the command name. | Function | Default keybinding | Description | |-----+-----+

--| | _complete_help | Ctrl+x h | displays information about context names, tags, and completion functions used when completing at the current cursor position || _complete_help | Alt+2 Ctrl+x h | as above but displays even more information || _complete_debug | Ctrl+x ? | performs ordinary completion, but captures in a temporary file a trace of the shell commands executed by the completion system * Gotchas (things to watch out for) Remember to include a #compdef line at the beginning of the file containing the completion function. Take care to use the correct type of quoting for specifications to _arguments or _regex_arguments: use double quotes if there is a parameter that needs to be expanded in the specification, single quotes otherwise, and make sure to use different quotes around item descriptions. Check that you have the correct number of ':'s in the correct places for specifications for _arguments, _alternative, _regex_arguments, etc. Remember to include an initial pattern to match the command word when using _regex_arguments (it does not need a matching action). Remember to put a null char '\$\0' at the end of any PATTERN argument for _regex_arguments * Tips Sometimes you have a situation where there is just one option that can come after a subcommand, and zsh will complete this automatically when tab is pressed after the subcommand. If instead you want it listed with its description before completing you can add another empty option (i.e. ':') to the ACTION like this ':TAG:DESCRIPTION:((opt1\: "description for opt1" \:))' Note this only applies to utility functions that use ACTIONS in their specification arguments (_arguments, _regex_arguments, etc.) * Other resources [[https://wikimatze.de/writing-zsh-completion-for-padrino/][Here]] is a nicely formatted short tutorial showing basic usage

of the `_arguments` function, and [[<https://web.archive.org/web/20190411104837/http://www.linux-mag.com/id/1106/>][here]] is a slightly more advanced tutorial using the `_arguments` function. [[<https://zsh.sourceforge.net/Doc/Release/Completion-System.html#Completion-System>][Here]] is the `zshcompsys` man page.