Hyprland

Related articles

Window manager

Wayland

Hyprland (https://hypr.land/) is an independent tiling Wayland compositor written in C++. Noteworthy features of Hyprland include dynamic tiling, tabbed windows, a clean and readable C++ code-base, and a custom renderer that provides window animations, rounded corners, and Dual-Kawase Blur on transparent windows. General usage and configuration is thoroughly documented at Hyprland wiki (https://wiki.hypr.land/).

1 Installation

Install one of the following:

- `hyprland (https://archlinux.org/packages/?name=hyprland)` - Latest upstream release.
- `hyprland-git (https://aur.archlinux.org/packages/hyprland-git/)`[AUR] - Latest commit to master branch.

As of #6608 (https://github.com/hyprwm/Hyprland/pull/6608), Hyprland uses `aquamarine (https://archlinux.org/packages/?name=aquamarine)` as its own rendering backend library. Before that, it bundled its own version of *wlroots*, which closely followed `wlroots-git (https://aur.archlinux.org/packages/wlroots-git/)`[AUR].

Note

- NVIDIA GPU users should also make sure to follow the NVIDIA specific page on the upstream Wiki (https://wiki.hypr.land/Nvidia/) before trying to launch Hyprland. Failure to do so will likely result in many bugs including not being able to log in, flashing windows and high CPU usage.
- Using an NVIDIA GPU with Hyprland is unsupported. Many users have had success but if something is broken then you are on your own.
- Make sure to install the Polkit package, or start and enable `seatd.service`. As the lack thereof will cause Hyprland to fail to start.
- For Vmware/VirtualBox users, it may be necessary to enable 3D acceleration in order to run Hyprland properly.

2 Configuration

Note

Most of the options listed (and more) are explained in detail on the upstream Wiki (https://wiki.hypr.land/Configuring/Variables/) variables page.

Configuration is done through a single configuration file, **hyprland.conf (https://github.com/hyprwm/Hyprland/blob/main/example/hyprland.conf)**, though it supports splitting the configuration into multiple files and including them in `hyprland.conf`. The default file is `/usr/share/hypr/hyprland.conf` and, after logging in for the first time, `~/.config/hypr/hyprland.conf`.

`hyprland.conf` includes directives to configure your devices (keyboards, mice, trackpads, monitors), as well as settings for animations, decorations, layout, etc. You can set key bindings, window rules, and execute commands (either once or each time) the configuration is reloaded.

The configuration is automatically reloaded each time you update the file. You can also use `hyprctl reload` for the same effect. For some settings (particularly input settings), you may have to restart your Hyprland session.

Settings can also be changed on the fly with *hyprctl* but they will not be saved.

2.1 Keyboard

2.1.1 Keymap

By default Hyprland will use `US Qwerty`, you can configure it as follows:

~/.config/hypr/hyprland.conf

```
# German Colemak layout
input {
    ...
    kb_layout = de
    kb_variant = colemak
    ...
    }
```

See **upstream's Wiki (https://wiki.hypr.land/Configuring/Variables/#input)** for all available options.

> **Note**
>
> Hyprland will override **locale** definitions so its necessary to change keymap if you do not use a `US` layout keyboard.

2.1.2 Typematic delay and rate

While Xorg users will be used to having this setting **defined at the server level**, on Wayland each compositor handles it on its own:

~/.config/hypr/hyprland.conf

```
# Repeat rate and delay
input {
    ...
    repeat_rate = 25
    repeat_delay = 600
    ...
    }
```

2.1.3 Keyboard backlight

Using keyboard brightness controls in Hyprland is possible. Install `brightnessctl` (https://archlinux.org/packages/?name=brightnessctl) then add the related binds (replace *keyboard_brightness_** with `SUPER, FX` or `XF86KbdBrightness` depending on how your hardware exposes the keyboard backlight):

~/.config/hypr/hyprland.conf

```
# Keyboard backlight
bind = , keyboard_brightness_up_shortcut, exec, brightnessctl -d *::kbd_backlight set +33%
bind = , keyboard_brightness_down_shortcut, exec, brightnessctl -d *::kbd_backlight set 33%-
```

It is also possible to have on-screen notifications that fire when changes are made.

2.1.4 Media keys

Using keyboard media controls in Hyprland is possible by making use of `XF86Audio` keysyms and an external application like `pavucontrol` (https://archlinux.org/packages/?name=pavucontrol) or `pamixer` (https://archlinux.org/packages/?name=pamixer) and playerctl:

~/.config/hypr/hyprland.conf

```
# Volume and Media Control
bind = , XF86AudioRaiseVolume, exec, pamixer -i 5
bind = , XF86AudioLowerVolume, exec, pamixer -d 5
bind = , XF86AudioMicMute, exec, pamixer --default-source -m
bind = , XF86AudioMute, exec, pamixer -t
bind = , XF86AudioPlay, exec, playerctl play-pause
bind = , XF86AudioPause, exec, playerctl play-pause
bind = , XF86AudioNext, exec, playerctl next
bind = , XF86AudioPrev, exec, playerctl previous
```

It is also possible to have on-screen notifications that fire when changes are made.

2.2 Touchpad gestures

Being a Wayland compositor, Hyprland has full support for touchpad gestures though they are disabled by default. To enable them, make the following edit:

~/.config/hypr/hyprland.conf

```
# Enable touchpad gestures
gesture = 3, horizontal, workspace
```

See the upstream Wiki (https://wiki.hypr.land/Configuring/Gestures/) for all available options.

2.3 Display settings

2.3.1 Screen sharing

See Screen-sharing (https://wiki.hypr.land/Useful-Utilities/Screen-Sharing/)

Being a wlroots-compatible compositor, Hyprland can utilize **xdg-desktop-portal-wlr (https://archlinux.org/packages/?name=xdg-desktop-portal-wlr)** to enable screen capture in a range of applications by way of **xdg-desktop-portal**.

Hyprland also maintains **xdg-desktop-portal-hyprland (https://archlinux.org/packages/?name=xdg-desktop-portal-hyprland)**, which supports screen sharing (including region sharing and window sharing), global shortcuts, and has a graphical picker utility. Usage of the portal is further documented in the **Hyprland wiki (https://wiki.hypr.land/Hypr-Ecosystem/xdg-desktop-portal-hyprland/)**.

It is worth noting that xdg-desktop-portal-hyprland does not include a file picker, for which users can additionally install **xdg-desktop-portal-gtk (https://archlinux.org/packages/?name=xdg-desktop-portal-gtk)**.

2.3.2 Setting screen resolution

Hyprland will try to detect your screen resolution automatically and then select either 1x, 1.5x, or 2x screen scaling. **[1] (https://github.com/hyprwm/Hyprland/blob/5ca48231287d67e75a3f21dbdbc47d6dc65752c4/src/helpers/Monitor.cpp#L563-L579)** However, in some cases it will fail and default to a fail-safe, usually if there are multiple screens present or if you have a hybrid laptop. If everything on your screen is huge then you need to configure your default monitor and resolution.

First find your default monitor using *hyprctl*:

```
$ hyprctl monitors
```

```
Monitor eDP-1 (ID 0):
        1920x1080@144.003006 at 0x0
        description: Chimei Innolux Corporation 0x153C (eDP-1)
        ...
```

Then add your monitor to the configuration:

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

```
...
# Monitor details
monitorv2 {
    output = eDP-1
    mode = 1920x1080@144
    position = 0x0
    scale = 1
}
...
```

`0x0` is a position offset used for multi screen setups and the final `1` is the screen scaling factor.

See the **upstream Hyprland Monitors Wiki (https://wiki.hypr.land/Configuring/Monitors/)** for more details.

2.3.3 Settings GUI

There is the `nwg-displays (https://archlinux.org/packages/?name=nwg-displays)` package, a GUI application for monitor arrangement, that supports Hyprland. It is part of the **nwg-shell** (but works standalone), see **nwg-displays github (https://github.com/nwg-piotr/nwg-displays)** for more details.

2.3.4 Screen backlight

Install `brightnessctl (https://archlinux.org/packages/?name=brightnessctl)` then add the following binds:

~/.config/hypr/hyprland.conf

```
# Screen brightness
bind = , XF86MonBrightnessUp, exec, brightnessctl s +5%
bind = , XF86MonBrightnessDown, exec, brightnessctl s 5%-
```

It is also possible to have **on-screen notifications** that fire when changes are made.

3 Usage

3.1 Starting

3.1.1 Universal Wayland Session Manager

Universal Wayland Session Manager wraps the compositor and accordingly configured applications and daemons through systemd unit files, allowing you to control them with *systemctl*.

Hyprland can be started via a **Display manager** with *uwsm* by selecting *hyprland (uwsm-managed)*.

You can start Hyprland with uwsm both in a **getty** via the following script in your **login shell**:

```
if uwsm check may-start && uwsm select; then
  exec systemd-cat -t uwsm_start uwsm start default
fi
```

Note

- `uwsm check may-start` checks whether it is OK to launch a wayland session, in particular if it is running from a **login shell**. However, you should still avoid using it inside `.bashrc` or other files which are sourced even by nonlogin shells.
- Hyprland no longer **recommends (https://github.com/hyprwm/hyprland-wiki/commit/7611480bc4c6a5cae1734cd7d6ea480b55e4c368)** starting your session with *uwsm* as it is considered experimental and intended for advanced users who understand its implications and **quirks (https://github.com/hyprwm/Hyprland/discussions/11255#discussioncomment-13938618)**.

Tip

If you want to bypass compositor selection menu and launch Hyprland directly, use this code in your **login shell**, instead:

```
if uwsm check may-start; then
  exec uwsm start hyprland.desktop
fi
```

Warning

If you choose to start Hyprland via *uwsm* then you should accordingly adjust your configuration. In particular:

- **you must avoid using the `exit` dispatcher or terminate the Hyprland process directly since this would interfer with the normal shutdown process**. Use instead `uwsm stop` or `loginctl terminate-user ""` to terminate Hyprland and exit user session, for example:

~/.config/hypr/hyprland.conf

```
bind = $mainMod, M, exec, uwsm stop
```

- Do not put environment variables in `hyprland.conf`, but use instead *uwsm* files `~/.config/uwsm/env` for variables common to all graphical sessions managed by uwsm (GTK, Qt, xcursor, ...) and `~/.config/uwsm/env-hyprland` for Hyprland exclusive environment variables (`HYPR*` and `AQ_*` variables for example). The format of these files is `export KEY=VALUE` on each line without comments. It is strongly suggested, if you use multiple GPUs, to put the environment variable `AQ_DRM_DEVICES` inside `env-hyprland` in order to avoid conflicts with other compositors.

Read the **related page in Hyprland's wiki (https://wiki.hypr.land/Useful-Utilities/Systemd-start/)** for additional information about configuration with uwsm.

3.1.2 Terminal

You can start Hyprland from a **getty** with the following command:

```
$ Hyprland
```

3.1.3 Display managers

While launching from a **display manager** is not officially supported, users have reported success launching from **GDM**, **SDDM**, and others. The **upstream wiki (https://wiki.hypr.land/Getting-Started/Master-Tutorial/#launching-hyprland)** maintains a compatibility list with display managers. The **`hyprland` (https://archlinux.org/packages/?name=hyprland)** package contains two **desktop entries**, and all Hyprland AUR packages will generate one automatically.

Both methods provide identical results, plus or minus a few environment variables and services.

3.1.4 Auto login

Users can automatically login by using a **display manager** or adapting the method described in **Xinit#Autostart X at login**.

3.2 hyprctl and IPC

hyprctl is a command line utility that comes installed with Hyprland to communicate with the display server. It allows you to dispatch commands to the server (equivalent to commands in the configuration file, but with a slightly different syntax), set keywords, send queries and request information. See the **full documentation (https://wiki.hypr.land/Configuring/Using-hyprctl/)**.

Hyprland also exposes **2 UNIX Sockets (https://wiki.hypr.land/IPC/)** for controlling and getting information about Hyprland via code or command-line utilities. These sockets broadcast events on focus change (windows, workspaces, monitors), creation of windows/workspace, and so on.

Both *hyprctl* and the IPC sockets can be effectively used in scripts to control Hyprland for complex tasks.

3.3 Autostart

When starting applications it is important to use the correct type of dispatcher, using `exec` incorrectly can result in applications being started multiple times taking up system resources and in the worst cases, causing a race condition that can crash your system.

> **Note**
>
> As mentioned in **#Configuration**, Hyprland automatically parses `hyprland.conf` **each time a change to the file is saved**: do not use `exec` for everything. In most cases you should use `exec-once` to launch applications and daemons at boot, as this command will not run again with a reload, only use `exec` if you are **absolutely sure** you want the command to run again on every reload.

> **Tip**
>
> If you started Hyprland via Universal Wayland Session Manager (uwsm) and the application you want to run at startup provides a systemd unit then you can **enable** it to automatically start it when Hyprland is ready. Otherwise, you can pass that application to `uwsm app` as argument in order to be managed by *uwsm*. For example:
>
> ~/.config/hypr/hyprland.conf
>
> ```
> exec-once = uwsm app -- mycommand --arg1 --arg2
> bind = SUPER, E, exec, uwsm app -- pcmanfm-qt.desktop
> ```

3.4 Setting environment variables

It is possible to set **environment variables** directly in `hyprland.conf` through the *env* keyword, which has a different syntax than the *env* UNIX command used by shells.

The differences are explained on the **upstream Wiki (https://wiki.hypr.land/Configuring/Environment-variables/)**.

4 Hypr-Ecosystem

> **Warning**
>
> Some of the tools outlined in the following section are still work in progress, as such bugs are to be expected. For this reason full instructions and examples will be omitted until they mature enough to be stable. Detailed instructions on their use is outlined on the **upstream Wiki (https://wiki.hypr.land/Hypr-Ecosystem/)**.

The Hyprland development team are building an ecosystem of applications tailored to be specifically used with Hyprland, these tools will include dispatchers allowing for them to be controlled with `hyprctl` rather than relying on scripts.

Currently the ecosystem consists of:

4.1 Hyprpaper

Hyprpaper is a wallpaper utility, it can be installed with the **`hyprpaper (https://archlinux.org/packages/?name=hyprpaper)`** package.

4.2 Hyprpicker

Hyprpicker is a utility to grab a colour from your desktop, it can be installed with the **`hyprpicker (https://archlinux.org/packages/?name=hyprpicker)`** package.

4.3 Hypridle

Hypridle is an idle management daemon, it can be installed with the **`hypridle (https://archlinux.org/packages/?name=hypridle)`** package.

4.4 Hyprlock

Hyprlock is a screen lock manager, it can be installed with the **`hyprlock (https://archlinux.org/packages/?name=hyprlock)`** package.

4.5 Hyprcursor

Hyprcursor is a new format for handling screen cursors that offers many improvements over the traditional way, it can be installed with the **`hyprcursor (https://archlinux.org/packages/?name=hyprcursor)`** package,

4.5.1 Hyprcursor themes

> **Tip**
>
> If you install hyprcursor and do not install a theme alongside it then it will fall back to your legacy cursor setting.

Cursor themes can be installed from the **AUR**, for example:

- `sweet-cursors-hyprcursor-git (https://aur.archlinux.org/packages/sweet-cursors-hyprcursor-git/)`[AUR]
- `nordzy-hyprcursors (https://aur.archlinux.org/packages/nordzy-hyprcursors/)`[AUR]
- `xcursor-pro-hyprcursor (https://aur.archlinux.org/packages/xcursor-pro-hyprcursor/)`[AUR]
- `hyprcursor-dracula-kde-git (https://aur.archlinux.org/packages/hyprcursor-dracula-kde-git/)`[AUR]

Instructions for porting existing themes to Hyprcursor are available on the **upstream GitHub repository (https://github.com/hyprwm/hyprcursor/tree/main/docs)**.

4.6 XDG-Desktop-Portal-Hyprland

Hyprland's own implementation of **XDG Desktop Portal**. Compatible with other wlroots-based compositors, but provides extra functionality when used on Hyprland. Available through the `xdg-desktop-portal-hyprland (https://archlinux.org/packages/?name=xdg-desktop-portal-hyprland)` package.

4.7 Hyprpolkitagent

Hyprpolkitagent is a **polkit** authentication daemon. It can be installed with the `hyprpolkitagent (https://archlinux.org/packages/?name=hyprpolkitagent)` package.

4.8 Hyprsunset

Hyprsunset is a small utility to provide a blue light filter for your system. It can be installed with the `hyprsunset (https://archlinux.org/packages/?name=hyprsunset)` package.

4.9 Hyprsysteminfo

Hyprsysteminfo is a system information fetching program like `neofetch (https://aur.archlinux.org/packages/neofetch/)`[AUR] or `fastfetch (https://archlinux.org/packages/?name=fastfetch)`. It can be installed with the `hyprsysteminfo (https://aur.archlinux.org/packages/hyprsysteminfo/)`[AUR] AUR package.

5 Tips and tricks

> **Note**
>
> - For all below sections there will usually be more than one way of achieving a similar result, everything provided here is a basic example.
> - For a comprehensive list of alternatives refer to **List of applications**, a Hyprland specific list can be found on the **upstream Wiki (https://wiki.hypr.land/Useful-Utilities/)**.

5.1 File manager

Hyprland requires a wayland-compatible external application if graphical file management is desired. Using **thunar (https://archlinux.org/packages/?name=thunar)** as an example, we simply need to assign it a keybind as follows:

~/.config/hypr/hyprland.conf

```
...
bind = SUPER, E, exec, thunar
...
```

5.2 Application launcher

Hyprland requires a wayland-compatible external application to launch applications. Using **wofi (https://archlinux.org/packages/?name=wofi)** as an example, we simply need to assign it a keybind as follows:

~/.config/hypr/hyprland.conf

```
...
bind = SUPER, F, exec, wofi --show drun
...
```

5.3 Idle

Hyprland requires a wayland-compatible external idle management daemon. The most common setup is **hypridle (https://archlinux.org/packages/?name=hypridle)** and **hyprlock (https://archlinux.org/packages/?name=hyprlock)**. You can lock your screen manually using a bind as follows:

~/.config/hypr/hyprland.conf

```
...
bind = SUPER, L, exec, hyprlock
...
```

5.3.1 Automatic screen locking and suspend

Create the following file:

~/.config/hypr/hypridle.conf

```
general {
    lock_cmd = pidof hyprlock || hyprlock
}

listener {
    timeout = 300
    on-timeout = loginctl lock-session
}

listener {
    timeout = 600
    on-timeout = systemctl suspend
}
```

> **Tip**
>
> You can adjust the timeout periods by editing the numerical values, in seconds. 300 is 5 minutes, 600 is 10 minutes etc.

Then run it:

~/.config/hypr/hyprland.conf

```
...
exec-once = hypridle
...
```

5.3.2 Turning off the screen using DPMS after a timeout period

Hyprland has a built in dispatcher to handle DPMS requests however using it as a direct keybind is not recommended, doing so will result in you not being able to turn the screen back on and will require you to reboot.

Edit the file from above and change it to read:

~/.config/hypr/hypridle.conf

```
general {
    lock_cmd = pidof hyprlock || hyprlock
}

listener {
    timeout = 300
    on-timeout = loginctl lock-session
}

listener {
    timeout = 600
    on-timeout = hyprctl dispatch dpms off
    on-resume = hyprctl dispatch dpms on
}

listener {
    timeout = 900
```

```
    on-timeout = systemctl suspend
}
```

5.4 Status bar

Hyprland requires a wayland-compatible external application to display a status bar. Using waybar (https://archlinux.org/packages/?name=waybar) as an example, we simply need to call it as follows:

~/.config/hypr/hyprland.conf

```
...
exec-once = waybar
...
```

5.4.1 Workspace overview

waybar (https://archlinux.org/packages/?name=waybar) has a built in, fully customisable module that supports Hyprland workspace switching natively.

See the waybar Wiki [2] (https://github.com/Alexays/Waybar/wiki/Module:-Workspaces) for details.

5.5 Polkit authentication

Polkit authentication requires the use of an external authentication agent. Hyprland recommends using hyprpolkitagent (https://archlinux.org/packages/?name=hyprpolkitagent) but any should work.

Call it as follows:

~/.config/hypr/hyprland.conf

```
...
exec-once = systemctl --user start hyprpolkitagent
...
```

5.6 Desktop wallpaper

Hyprland requires a wayland-compatible external application to manage desktop wallpapers. Using hyprpaper (https://archlinux.org/packages/?name=hyprpaper) as an example, we simply need to call it as follows:

~/.config/hypr/hyprland.conf

```
...
exec-once = hyprpaper
...
```

Additionally since hyprpaper (https://archlinux.org/packages/?name=hyprpaper) requires a configuration file to start; make the file as follows:

~/.config/hypr/hyprpaper.conf

```
preload = /home/me/amongus.png
wallpaper = monitor, /home/me/amongus.png
```

Replace *monitor* with the monitor you would like the wallpaper to be set on, you can grab a list via `hyprctl monitors`.

5.6.1 Using a script to randomize the wallpaper

Create the following script and make sure its **executable**:

~/.config/hypr/scripts/hyprpaper-random

```
#!/usr/bin/env bash

WALLPAPER_DIR="$HOME/.config/hypr/wallpapers/"
CURRENT_WALL=$(hyprctl hyprpaper listloaded)

# Get a random wallpaper that is not the current one
WALLPAPER=$(find "$WALLPAPER_DIR" -type f ! -name "$(basename "$CURRENT_WALL")" | shuf -n 1)

# Apply the selected wallpaper
hyprctl hyprpaper reload ,"$WALLPAPER"
```

Next create a new *directory* to store wallpapers, something like `~/.config/hypr/wallpapers` should work fine, and populate it with any images you want.

Finally call the script when the *specified* bind is pressed:

~/.config/hypr/hyprland.conf

```
...
$mainMod = super

bind = $mainMod, r, exec, ~/.config/hypr/scripts/hyprpaper-random
...
```

5.7 On screen notifications

On screen notifications for actions like brightness and volume changes are possible by using external notification daemons. This is a very complex topic and covering it completely is beyond the scope of this page. Rather, this section will focus on **mako (https://archlinux.org/packages/?name=mako)** so go ahead and **install** it.

See **Desktop notifications** for further instructions and **Desktop notifications#Standalone** for a list of alternatives.

> **Note**
>
> - All scripts offered here are examples and will very likely need to be adjusted for your setup.
> - All scripts in this section must be **executable**.

5.7.1 Mako

Mako is a lightweight notification daemon, you can read **mako(5) (https://man.archlinux.org/man/mako.5)** for details. Its configuration file is `~/.config/mako/config`, icons used for OSD are stored at `~/.config/mako/icons/` and should be in PNG format.

For the rest of this section all the images used by the scripts are available from **this GitHub folder (https://github.com/SolDoesTech/HyprV4/tree/main/HyprV/mako/icons)**.

5.7.1.1 Keyboard backlight notifications

First create the following script:

~/.config/hypr/scripts/kbbacklight

```
#!/usr/bin/env bash

iDIR="$HOME/.config/mako/icons"

# Get brightness
get_backlight() {
        LIGHT="$(cat /sys/class/leds/*::kbd_backlight/brightness)"
        echo "${LIGHT}"
}

# Get icons
get_icon() {
        current="$(cat /sys/class/leds/*::kbd_backlight/brightness)"

        if [[ ("$current" -ge "0") && ("$current" -le "1") ]]; then
                icon="$iDIR/brightness-20.png"
        elif [[ ("$current" -ge "1") && ("$current" -le "2") ]]; then
                icon="$iDIR/brightness-60.png"
        elif [[ ("$current" -ge "2") && ("$current" -le "3") ]]; then
                icon="$iDIR/brightness-100.png"
        fi
}

# Notify
notify_user() {
        notify-send -h string:x-canonical-private-synchronous:sys-notify -u low -i "$icon" "Keyboard Brightn
ess : $(brightnessctl -d '*::kbd_backlight' g)"
}

# Increase brightness
inc_backlight() {
        brightnessctl -d *::kbd_backlight set 33%+ && get_icon && notify_user
}

# Decrease brightness
dec_backlight() {
        brightnessctl -d *::kbd_backlight set 33%- && get_icon && notify_user
}

# Zero brightness
zero_backlight() {
        brightnessctl -d *::kbd_backlight s 0%
}

# Full brightness
full_backlight() {
        brightnessctl -d *::kbd_backlight s 100%
}

# Execute accordingly
if [[ "$1" == "--get" ]]; then
        brightnessctl -d '*::kbd_backlight' g
elif [[ "$1" == "--inc" ]]; then
```

```
        inc_backlight
elif [[ "$1" == "--dec" ]]; then
        dec_backlight
elif [[ "$1" == "--zero" ]]; then
        zero_backlight
elif [[ "$1" == "--full" ]]; then
        full_backlight

else
        get_backlight
fi
```

Then add a new bind, or edit any **existing one**:

~/.config/hypr/hyprland.conf

```
# Keyboard brightness
bind = keyboard_brightness_up_shortcut, exec, ~/.config/hypr/scripts/kbbacklight --inc
bind = keyboard_brightness_down_shortcut, exec, ~/.config/hypr/scripts/kbbacklight --dec
```

5.7.1.2 Media key notifications

First create the following script:

~/.config/hypr/scripts/volume

```
#!/usr/bin/env bash

iDIR="$HOME/.config/mako/icons"

# Get Volume
get_volume() {
        volume=$(pamixer --get-volume)
        echo "$volume"
}

# Get icons
get_icon() {
        current=$(get_volume)
        if [[ "$current" -eq "0" ]]; then
                echo "$iDIR/volume-mute.png"
        elif [[ ("$current" -ge "0") && ("$current" -le "30") ]]; then
                echo "$iDIR/volume-low.png"
        elif [[ ("$current" -ge "30") && ("$current" -le "60") ]]; then
                echo "$iDIR/volume-mid.png"
        elif [[ ("$current" -ge "60") && ("$current" -le "100") ]]; then
                echo "$iDIR/volume-high.png"
        fi
}

# Notify
notify_user() {
        notify-send -h string:x-canonical-private-synchronous:sys-notify -u low -i "$(get_icon)" "Volume :
$(get_volume) %"
}

# Increase Volume
inc_volume() {
        pamixer -i 5 && notify_user
}

# Decrease Volume
dec_volume() {
        pamixer -d 5 && notify_user
}

# Toggle Mute
toggle_mute() {
```

```
        if [ "$(pamixer --get-mute)" == "false" ]; then
                pamixer -m && notify-send -h string:x-canonical-private-synchronous:sys-notify -u low -i "$i
DIR/volume-mute.png" "Volume Switched OFF"
        elif [ "$(pamixer --get-mute)" == "true" ]; then
                pamixer -u && notify-send -h string:x-canonical-private-synchronous:sys-notify -u low -i
"$(get_icon)" "Volume Switched ON"
        fi
}

# Toggle Mic
toggle_mic() {
        if [ "$(pamixer --default-source --get-mute)" == "false" ]; then
                pamixer --default-source -m && notify-send -h string:x-canonical-private-synchronous:sys-not
ify -u low -i "$iDIR/microphone-mute.png" "Microphone Switched OFF"
        elif [ "$(pamixer --default-source --get-mute)" == "true" ]; then
                pamixer -u --default-source u && notify-send -h string:x-canonical-private-synchronous:sys-n
otify -u low -i "$iDIR/microphone.png" "Microphone Switched ON"
        fi
}
# Get icons
get_mic_icon() {
        current=$(pamixer --default-source --get-volume)
        if [[ "$current" -eq "0" ]]; then
                echo "$iDIR/microphone.png"
        elif [[ ("$current" -ge "0") && ("$current" -le "30") ]]; then
                echo "$iDIR/microphone.png"
        elif [[ ("$current" -ge "30") && ("$current" -le "60") ]]; then
                echo "$iDIR/microphone.png"
        elif [[ ("$current" -ge "60") && ("$current" -le "100") ]]; then
                echo "$iDIR/microphone.png"
        fi
}
# Notify
notify_mic_user() {
        notify-send -h string:x-canonical-private-synchronous:sys-notify -u low -i "$(get_mic_icon)" "Mic-Le
vel : $(pamixer --default-source --get-volume) %"
}

# Increase MIC Volume
inc_mic_volume() {
        pamixer --default-source -i 5 && notify_mic_user
}

# Decrease MIC Volume
dec_mic_volume() {
        pamixer --default-source -d 5 && notify_mic_user
}

# Execute accordingly
if [[ "$1" == "--get" ]]; then
        get_volume
elif [[ "$1" == "--inc" ]]; then
        inc_volume
elif [[ "$1" == "--dec" ]]; then
        dec_volume
elif [[ "$1" == "--toggle" ]]; then
        toggle_mute
elif [[ "$1" == "--toggle-mic" ]]; then
        toggle_mic
elif [[ "$1" == "--get-icon" ]]; then
        get_icon
elif [[ "$1" == "--get-mic-icon" ]]; then
        get_mic_icon
elif [[ "$1" == "--mic-inc" ]]; then
        inc_mic_volume
elif [[ "$1" == "--mic-dec" ]]; then
        dec_mic_volume
else
        get_volume
fi
```

Then add the following (or edit any existing binds):

~/.config/hypr/hyprland.conf

```
# Volume
bind = , XF86AudioRaiseVolume, exec, ~/.config/hypr/scripts/volume --inc
bind = , XF86AudioLowerVolume, exec, ~/.config/hypr/scripts/volume --dec
bind = , XF86AudioMicMute, exec, ~/.config/hypr/scripts/volume --toggle-mic
bind = , XF86AudioMute, exec, ~/.config/hypr/scripts/volume --toggle
```

5.7.1.3 Screen backlight notifications

First create the following script:

~/.config/hypr/scripts/backlight

```
#!/usr/bin/env bash

iDIR="$HOME/.config/mako/icons"

# Get brightness
get_backlight() {
        LIGHT=$(printf "%.0f\n" $(brightnessctl g))
        echo "${LIGHT}"
}

# Get icons
get_icon() {
        current="$(get_backlight)"
        if [[ ("$current" -ge "0") && ("$current" -le "19200") ]]; then
                icon="$iDIR/brightness-20.png"
        elif [[ ("$current" -ge "19200") && ("$current" -le "38400") ]]; then
                icon="$iDIR/brightness-40.png"
        elif [[ ("$current" -ge "38400") && ("$current" -le "57600") ]]; then
                icon="$iDIR/brightness-60.png"
        elif [[ ("$current" -ge "57600") && ("$current" -le "76800") ]]; then
                icon="$iDIR/brightness-80.png"
        elif [[ ("$current" -ge "76800") && ("$current" -le "96000") ]]; then
                icon="$iDIR/brightness-100.png"
        fi
}

# Notify
notify_user() {
        notify-send -h string:x-canonical-private-synchronous:sys-notify -u low -i "$icon" "Brightness : $(g
et_backlight)"
}

# Increase brightness
inc_backlight() {
        brightnessctl s +5% && get_icon && notify_user
}

# Decrease brightness
dec_backlight() {
        brightnessctl s 5%- && get_icon && notify_user
}

# Execute accordingly
if [[ "$1" == "--get" ]]; then
        get_backlight
elif [[ "$1" == "--inc" ]]; then
        inc_backlight
elif [[ "$1" == "--dec" ]]; then
        dec_backlight
else
        get_backlight
fi
```

Then add the following (or edit any existing binds):

~/.config/hypr/hyprland.conf

```
# Screen brightness
bind = , XF86MonBrightnessUp, exec, ~/.config/hypr/scripts/backlight --inc
bind = , XF86MonBrightnessDown, exec, ~/.config/hypr/scripts/backlight --dec
```

5.7.1.4 Keyboard language notifications

To run this script, you need a command-line JSON processor **gojq (https://aur.archlinux.org/packages/gojq/)**[AUR].

First create the following script:

~/.config/hypr/scripts/lang

```
#!/usr/bin/env bash

icon="$HOME/.config/mako/icons/language.png"

# Get language
get_lang() {
        lang=$(hyprctl devices -j | gojq -r '.keyboards[] | select(.name == "at-translated-set-2-keyboard")
| .active_keymap' | cut -c 1-2 | tr 'A-Z' 'a-z')
        case $lang in
                en)
                        lang="English language"
                        ;;
                ru)
                        lang="Русский язык"
                        ;;
                uk)
                        lang="Українська мова"
                        ;;
        esac
        echo $lang
}

# Notify
notify-send -h string:x-canonical-private-synchronous:sys-notify -u low -i "$icon" "$(get_lang)"
```

Then add the following (or edit any existing binds):

~/.config/hypr/hyprland.conf

```
device:at-translated-set-2-keyboard {
    kb_layout = us,ru,ua
    kb_variant = lang
    kb_options = grp:win_space_toggle
}

# Language
bind = SUPER, SPACE, exec, ~/.config/hypr/scripts/lang
```

5.8 Power control

Hyprland requires a wayland-compatible external application for power control. Using **nwg-bar (https://archlinux.org/packages/?name=nwg-bar)** as an example, we simply need to bind it as follows:

~/.config/hypr/hyprland.conf

```
...
bind = SUPER, ESCAPE, exec, nwg-bar
...
```

5.9 Clipboard

Wayland clipboard behaviour deletes data when closing the application we copied it from. Other desktop environments work around this by using dedicated clipboard managers and on Hyprland there are multiple compatible choices. See the upstream Wiki (https://wiki.hypr.land/Useful-Utilities/Clipboard-Managers/) for more information.

This section will cover `cliphist` (https://archlinux.org/packages/?name=cliphist) as it supports copying images as well as text, start by adding the following:

~/.config/hypr/hyprland.conf

```
...
exec-once = wl-paste --type text --watch cliphist store
exec-once = wl-paste --type image --watch cliphist store
...
```

Then create a bind to call the history in your chosen application launcher:

~/.config/hypr/hyprland.conf

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

Now pressing `Super+v` will open up a `wofi` window with a clipboard history list.

5.10 Enable/disable devices

To enable/disable devices (e.g. touchpad), first use:

```
$ hyprctl devices
```

to get the name of your device.

Put these lines of code into your configuration file (replace <device_name> with the name of your device queried above) to turn the device on/off:

~/.config/hypr/hyprland.conf

```
device {
  name = <device_name>
  enabled = {true/false}
}
```

To dynamically switch the device on/off use hyprctl:

```
$ hyprctl keyword "device[<device_name>]:enabled" {true|false}
```

You can also create a keybinding, e.g.:

~/.config/hypr/hyprland.conf

```
...
bind = $mainMod, t, exec, hyprctl keyword "device[pixa3854:00-093a:0274-touchpad]:enabled" false
bind = $mainMod Shift, t, exec, hyprctl keyword "device[pixa3854:00-093a:0274-touchpad]:enabled" true
...
```

Note: Prior to Hyprland v0.34(?), the following legacy syntax was used:

`device:<device_name>:enabled`

This older format has been removed. Also, earlier configuration files did not use a block-based `device { name = <device_name> ... }` structure, but a `device:<device_name> { ... }` structure.

5.11 Separate dconf profile

In case you do not want to poison settings for other GTK-based DEs, you can use a separate `dconf` profile. For example:

Declare new global dconf profile:

/etc/dconf/profile/hyprland

```
user-db:hyprland
```

~/.config/hypr/hyprland.conf

```
...
env = DCONF_PROFILE, hyprland
...
```

Now you can use *gsettings* and it should not affect other desktop environments.

6 Troubleshooting

6.1 Native (wayland) electron apps flickering on NVIDIA

It is a widespread issue among NVIDIA users on Hyprland **[3] (https://github.com/hyprwm/Hyprland/issues/6701)**, **[4] (https://github.com/hyprwm/Hyprland/issues/6703)** because of lack of support for explicit sync in Hyprland **[5] (https://github.com/hyprwm/Hyprland/issues/4857)**. Recommended temporary fix is using X11 (XWayland) with the problematic apps by passing them `--ozone-platform-hint=x11`, or setting `env = ELECTRON_OZONE_PLATFORM_HINT,x11` in `~/.config/hypr/hyprland.conf` to force all electron apps to run using XWayland.

6.2 Jetbrains apps focus issues

Jetbrains apps (Pycharm, Intellij) can have strange focus problems such as:

- Unable to drag tab from the tab bar [6] (https://github.com/hyprwm/Hyprland/issues/1120) to either a split, or another tab stack without focus being stolen and the tab being dropped as soon as you drag it past the current tab bar.
- Autocomplete popup window stealing focus until the mouse is moved.

To mitigate the issue add this to *hyprlands* configuration file:

~/.config/hypr/hyprland.conf

```
windowrulev2 = noinitialfocus,xwayland:1
```

7 See also

- Hyprland Website (https://hypr.land/)
- The official documentation (https://wiki.hypr.land/)
- Hyprland Github Page (https://github.com/hyprwm/Hyprland/)
- Community-maintained list of tools, plugins and extensions (https://github.com/hyprland-community/awesome-hyprland)
- SolDoesTech Github Repo - Notification scripts creator (https://github.com/SolDoesTech/HyprV4)