SwayWM — сам себе UnixPorn

11 мин 60K

Настройка Linux*

Туториал

Всем привет. В этой статье я опишу свой опыт настройки и использования sway — тайлингового оконного менеджера для Linux.

Что это такое и зачем оно нужно?

Официально, sway — тайлинговый оконный менеджер, прозрачная замена i3wm, работающая поверх Wayland.

> Вейланд, он же Вёлунд и т.п. — персонаж из древнегерманской и древнескандинавской мифологии, книгу о которой в своё время написал английский фольклорист Джесси Вестон (Jessie Weston). Теперь вы знаете.

Однако, на мой взгляд, главная особенность sway — в том, что он является конструктором, со всеми преимуществами и недостатками такого подхода. Если вам нравятся Vim (сходство с ним усугубляется ориентацией на использование клавиатуры), Archlinux и подобные проекты, то советую присмотреться и к sway. Лично моё мнение — один раз потратив чуть больше времени на его настройку, вы получите очень стабильное и удобное лично для вас окружение, в котором не будет происходить ничего неожиданного.

Если недостатки sway и Wayland (такие как отсутствие поддержки закрытых драйверов NVidia и необходимость искать аналоги для многих привычных программ) кажутся вам фатальными, то советую обратить внимание на иксовый i3wm. Части конструктора там будут совсем другими, придётся ставить xinit, настраивать разнообразные .Xauthority, возможно бороться с тирингом, но общая логика работы системы — точно такая же.

Установка

Sway наверняка есть в репозиториях вашего дистрибутива. Однако, если вам хочется установить более новую версию, то это очень просто. На странице проекта на github приведён куцый список его зависимостей. Установите их, склонируйте репозиторий, склонируйте `wlroots` в папку `subprojects` :

```
hub clone swaywm/sway
hub clone swaywm/wlroots sway/subprojects/wlroots
```

(тут я использую hub. Нестандартный, но очень удобный инструмент). После этого можно конфигурировать и собирать:

```
meson setup ./sway-build ./sway --buildtype=minsize
ninja -C ./sway-build
doas ninja -C ./sway-build install
```

> doas — легковесная замена sudo от разработчиков OpenBSD. В Linux как правило используют её форк OpenDoas

Запуск

Если вы хотите запускать sway из командной строки, не пользуясь менеджерами вроде SDDM, то единственный правильный способ делать это —

```
$ exec sway
```

Если вы запустите sway без `exec`, заблокируете экран с помощью swaylock и sway упадёт, то вас выкинет обратно в командную строку. Блокировка окажется бесполезной.

> Нелюбителям systemd на заметку: sway не требуется ни systemd-logind, ни elogind. Правда, в этом случае придётся установить suid-бит или настроить capabilities. См. wiki.

Переменные окружения

Есть несколько мест, куда я их прописываю.

Shell-скрипт по адресу `~/.local/bin/sway` — для переменных, специфичных для sway. Например я хочу, чтобы Qt-приложения не показывали декорации и использовали тему из `qt5ct`:

```
#!/bin/bash

QT_QPA_PLATFORMTHEME=qt5ct \
```

```
QT_WAYLAND_DISABLE_WINDOWDECORATION=1 \
/usr/bin/sway
```

`~/.pam_environment` — для переменных, которые я хочу сразу сделать доступными в пользовательской сессии systemd. Пример:

```
SSH_AUTH_SOCK DEFAULT="${XDG_RUNTIME_DIR}/gnupg/S.gpg-agent.ssh"
```

Однако меняйте этот файл с осторожностью, особенно если вы используете и другие окружения. Например, KDE Plasma очень не понравится, если оно обнаружит там `$WAYLAND_DISPLAY` (по иронии, заведующий этим файлом модуль `pam_env` был создан N лет назад в первую очередь для того, чтобы устанавливать переменную `$DISPLAY`).

`~/.bashrc` — для всех остальных случаев.

Настройка sway

Ввод

В моей системе всё совсем просто: устанавливаем русскую и английскую раскладки и переключаемся между ними по `Alt+Shift`. В конфиге sway это выглядит так:

```
input * {
    xkb_layout us,ru
    xkb_options grp:alt_shift_toggle
}
```

Sway использует для ввода библиотеку `libinput`, ту же самую, что KDE и Gnome. А значит, нам доступно множество её опций. Настройки тачпада, естественные прокрутки, ускорения указателей мыши, и т.д. и т.п. Список опций можно изучить в `man sway-input`, а список устройств ввода — в выводе команды

```
swaymsg -t get_inputs
```

Однако я хотел бы обратить внимание на одну интересную возможность. Если вы переезжаете с иксового окружения, то можете просто перенести оттуда свои настройки

клавиатуры. Выполните в этом окружении команду

```
$ xkbcomp $DISPLAY /path/to/keymap.xkb
```

а в конфиге sway пропишите

```
input * {
    xkb_file /path/to/keymap.xkb
}
```

Хоткеи

Хоткеи, как и все остальное в sway, задаются в конфиге. Есть 3 команды для их определения:

1. `bindswitch` . Выполняется при открытии/закрытии крышки ноутбука, переходе в планшетный режим, и т.п.
2. `bindcode` . Выполняется при нажатии клавиши с заданным кодом. Не зависит от текущей раскладки, особенно полезно для мультимедийных клавиш.
3. `bindsym` . Текущий введенный символ, зависит от раскладки. Однако, при добавлении параметра `--to-code` этот символ будет неявно преобразован в соответствующий код для первой раскладки из `xkb_layout` (см. настройки ввода).

Например, такой хоткей прибьёт текущее окно независимо от раскладки:

```
bindsym --to-code $mod+Shift+q kill
```

Подробнее про эти параметры можно почитать в мане: `man 5 sway` .

Чтобы легко определять, что именно было нажато, автор sway написал специальный кейлоггер (ему требуются права рута для работы, так что за безопасность можно не беспокоиться).

Вывод

Список доступных устройств вывода можно получить командой

```
swaymsg -t get_outputs
```

В конфиге для каждого из них можно задать видеорежим, масштабирование (в т.ч. дробное), картинку, поворот и т.п. Простейшая конфигурация выглядит так:

```
output HDMI-A-1 mode 1920x1080@60Hz
output * bg /path/to/wallpaper.jpg fill
```

Полный список опций можно посмотреть в мане `man sway-output`.
Также есть программа wlr-randr, в которой всё это можно задавать из командной строки (то есть, это аналог XRandr для Wayland).

Swaybar

Swaybar — панель, на которой показывается статусная информация. Дата, время, системный лоток, и т.д. и т.п. Обычно фанаты i3 и sway проводят больше всего времени за настройкой именно этой панели (или панелей, их может быть несколько). Лично мне хватает такого, близкого к минимальному конфига:

```
bar {
    position top
    colors {
        statusline #ffffff
        background #282828E6
        inactive_workspace #282936BF #282936BF #5c5c5c
    }
    font Hack 11
    status_command i3blocks
}
```

Есть множество опций настройки положения, цветов, шрифтов, трея и т.п. Почитать о них можно в мане — `man 5 sway-bar`.

Swaybar занимается тем, что читает и парсит json в формате i3bar, выводимый `status_command`. Есть множество программ, которые могут тут использоваться. i3status, waybar, i3status-rs, и множество других проектов. Лично я предпочитаю i3blocks из-за простоты конфига и лёгкости добавления собственных блоков.

Настройка прикладных программ

Уведомления

Этим может заниматься демон mako (скорее всего есть в вашем дистрибутиве). Там тоже можно настраивать цвета и т.п., но это всё опционально. Просто запустите его, и он будет работать.

Mako реализует использующую dbus спецификацию XDG Desktop Notifications. Её поддерживают многие программы, в том числе Chromium, Firefox и Telegram.

Эмулятор терминала

В окружениях вроде KDE Plasma обычно пользуются встроенными эмуляторами, показывающими меню, вкладки, и множество других элементов оформления. Но зачем всё это в sway? Я предпочитаю Alacritty — очень быстрый благодаря использованию GPU эмулятор, кстати написанный на Rust. Я добавил такие хоткеи в секцию `key_bindings` в его конфиге (`~/.config/alacritty/alacritty.yml`):

```
- { key: T, mods: Control|Shift, action: SpawnNewInstance }
```

открыть новое окно с текущим каталогом. Прекрасная замена вкладкам, а о расположении позаботится sway.

```
- { key: Up,    mods: Control|Shift, action: ScrollLineUp,    mode: ~Alt }
- { key: Down,  mods: Control|Shift, action: ScrollLineDown,  mode: ~Alt }
```

прокрутить на строку вверх/вниз.

Пишем в конфиге sway

```
set $term alacritty
bindsym $mod+Return exec $term
```

Alacritty не поддерживает лигатуры, так как они влияют на скорость рендеринга. Если они вам всё-таки нужны, обратите внимание на kitty — этот эмулятор тоже очень быстр, а ещё там под капотом просто пугающее количество фич и настроек.

Запуск приложений

Приложения в sway обычно запускаются с помощью вызываемого по хоткею меню. В конфиге пишут что-то вроде

```
set $menu ...
bindsym --to-code $mod+d exec $menu
```

Для показа меню есть множество программ. Некоторым, таким как dmenu или rofi, нужен XWayland. Среди работающих под Wayland можно выделить bemenu и даже krunner (то, что появляется в KDE по `Alt+F2`. Да, это независимая программа).

Однако я предпочитаю более универсальное решение, а именно sway-launcher-desktop. Несмотря на название, оно не зависит от sway и вообще является не GUI-программой, а работающим в консоли небольшим shell-скриптом. Этот скрипт формирует меню из программ в `$PATH`, установленных desktop-файлов (в соответствии со спецификацией XDG Desktop Entry) и возможно дополнительных источников (настраиваются в конфиге). Для показа и поиска по меню используется замечательная утилита `fzf` — обратите на неё внимание.

Таким образом, для показа меню нам надо открывать окно терминала с этой программой. С alacritty это делается так:

```
set $menu exec alacritty -e sway-launcher-desktop
bindsym --to-code $mod+d exec $menu
```

Возможно, вам захочется сделать это окно плавающим по умолчанию. В i3 и sway подобные правила настраиваются в конфиге при помощи директивы `for_window`. Для сопоставления правила с окном `sway-launcher-desktop` ему надо назначить какой-нибудь уникальный класс, у меня это просто `Launcher`. Имейте ввиду, что из-за использования Wayland перечисленные в мануале i3 критерии не подходят для sway. Правильные можно найти в `man 5 sway`, секция `CRITERIA`. В данном случае вполне подойдёт `app_id`, и итоговый конфиг выглядит так:

```
set $menu exec alacritty --class Launcher -e sway-launcher-desktop
for_window [app_id="Launcher"] floating enable, border pixel 10, sticky enable
bindsym --to-code $mod+d exec $menu
```

Автозапуск приложений

В принципе, для автозапуска программы можно просто написать

```
exec /path/to/program
```

в конфиге sway, или даже создать пользовательский юнит systemd. Однако, в этой статье я воспользуюсь принятой в KDE и Gnome спецификацией XDG Autostart.

Спецификация заключается в запуске desktop-файлов, расположенных в каталогах `/etc/xdg/autostart` и `~/.config/autostart`. Как правило, эти файлы поставляются в пакетах с программами, но их легко создать и самому.

Здесь проявляется модульная сущность sway. Sway сам по себе ничего не знает ни про XDG Autostart, ни про XDG Desktop Entry. Больше того, его разработчики активно не любят эти и все подобные спецификации. Однако, с ними умеет управляться консольная утилита dex (скорее всего, есть в репозитории вашего дистрибутива).

Чтобы проверить, что будет в автозапуске, наберите команду

```
$ dex -ade Sway
```

`a` значит autostart, `d` — "dry run", `-e Sway` задаёт название окружения (может быть любым) и применяется для фильтрации.

Скорее всего, получившийся список вас не устроит. Например, у меня там оказался `kgpg`. Поступаем следующим образом: копируем соответствующий desktop-файл из `/etc/xdg/autostart` в `~/.config/autostart`, добавляем туда строку

```
NotShowIn=Sway
```

и снова проверяем автозапуск. Многим программам (особенно написанным на Electron) может не понравиться запускаться под Wayland. Поступаем с ними точно так же. Копируем desktop-файл и меняем там строчку запуска. Было:

```
Exec=/usr/bin/skypeforlinux
```

стало:

```
Exec=env GDK_BACKEND=x11 /usr/bin/skypeforlinux
```

этот же приём можно применять и для sway-launcher-desktop из предыдущего пункта. Когда все проблемы решены, добавляем в конфиг sway строчку

```
exec dex -ae Sway
```

Либо можно поступать наоборот: копировать нужные desktop-файлы в какой-нибудь нестандартный каталог и натравливать на него dex командой

```
exec dex -as /path/to/your/dir
```

Скриншоты

В этом нам помогут двое из ларца. slurp позволяет выбрать произвольную прямоугольную область на экране, а grim делает из неё скриншот. В моём конфиге это выглядит так:

```
bindsym Print exec grim ~/Pictures/screen-"$(date +%s)".png
bindsym $mod+Print exec grim -g "$(slurp)" ~/Pictures/screen-"$(date +%s)".png
```

`$mod+PrintScreen` — скриншот произвольной области, `PrintScreen` — скриншот всего экрана.

Скринкасты

Если вам нужно записать скринкаст в файл или постримить его в RTMP-поток (например на Twitch), то с этим поможет wf-recorder (тоже поддерживает slurp).

Если хотите использовать OBS Studio, то для этого есть плагин wlrobs.

Однако самое универсальное решение, поддерживаемое с одной стороны в KDE и Gnome, а с другой, в частности, в браузерах Chrome и Firefox — это использовать Pipewire и спецификацию XDG Desktop Portal. Для sway есть нужная прослойка xdg-desktop-portal-wlr, правда я не проверял её работоспособность.

Ночной режим

Ночной режим заключается в замене цветов монитора на более тёплые в тёмное время суток. Ученые сомневаются, насколько это на самом деле полезно, а разработчики Wayland решили, что стандартизировать протокол гамма-коррекции бесполезно по техническим причинам.

Тем не менее, мне эта фича нравится, и в sway она тоже есть. Для ночного режима нам понадобится вот этот форк redshift с поддержкой специфичного для sway протокола.

Устанавливаем его любым способом, в конфиге `~/.config/redshift/redshift.conf` выбираем нужный протокол и указываем долготу и широту.

```
[redshift]
location-provider=manual
adjustment-method=wayland

[manual]
lat=xx.xx
lon=yy.yy
```

Блокировка и отключение экрана

Здесь приходят на помощь двое из другого ларца. `swayidle` запускает заданные команды по таймауту в случае отсутствия активности (или наоборот, её появления), а `swaylock` блокирует экран и требует ввести пароль. Отключить экран (или все экраны, если их несколько) можно командой

```
swaymsg "output * dpms off"
```

Вот как будет выглядеть демон, блокирующий экран через 300 секунд неактивности и отключающий его через 600:

```
swayidle -w \
    timeout 300 'swaylock -f -c 000000' \
    timeout 600 'swaymsg "output * dpms off"' \
         resume 'swaymsg "output * dpms on"' \
    before-sleep 'swaylock -f -c 000000'
```

Запустите его любым способом.

i3blocks

Готовые блоки с календарем, сетью, состояние диска и т.д. и т.п. можно позаимствовать из репозитория i3blocks-contrib. В i3blocks они добавляются в конфиге по адресу `~/.config/i3blocks/config`. Например, блок с обновляемыми раз в 5 секунд днём недели, датой и временем:

```
[time]
command=date +"%a %d/%m %H:%M"
interval=5
```

Но это был слишком простой пример. На самом деле, у любителей i3 и sway есть священный грааль: показ заголовка текущего окна в верхней панели. Давайте разберёмся, как этого добиться.

Прежде всего начинаем читать `man sway-ipc` и обнаруживаем, что там можно подписываться на события. Нужное имеет код `0x80000003. WINDOW` и возвращает json-объекты такой структуры:

```
{
  "change" : "focus | title | ...",
  "container": {
    focused: true | false,
    name: "...",
    ...
  }
}
```

Действительно, либо само окно может сменить свой заголовок (`change == title`), либо мы можем переместить фокус на другое окно (`change == focus`). Однако, окно может сменить заголовок и не под фокусом. Чтобы отфильтровать такие события, мы должны проверять свойство `container.focused` .

Можно описать всю эту логику на Python или Go, однако есть способ и получше. `swaymsg` позволяет легко подписываться на нужные события и выводить их в stdout:

```
$ swaymsg -m -t SUBSCRIBE "['window']"
```

а получающийся на выходе json мы будем обрабатывать в `jq`. Не вдаваясь в тонкости синтаксиса запросов этой замечательной утилиты, вот что получается в итоге:

```
$ query='select(.change == "focus" or (.change == "title" and .container.focuse
$ swaymsg -m -t SUBSCRIBE "['window']" | jq --unbuffered -r "$query"
```

Скрипт из двух строчек на баше, и больше никакого кода! Флаг `--unbuffered` нужен, иначе i3blocks будет получать новые строки не сразу, а только при очистке буфера.

Наконец, в конфиг i3blocks надо добавить такой блок:

```
[active_window]
command=/path/to/our/script.sh
interval=persist
```

Домашнее задание: научите grim делать скриншот текущего окна. Говорят, что такие вещи невозможны в Wayland, но мужики сомневаются.

Домашнее задание 2: научите i3blocks показывать текущую раскладку клавиатуры в виде флага. В i3blocks-contrib есть пара блоков, но они показывают текст вместо флага и не работают в Wayland.

Прочие программы

wl-clipboard позволяет работать с буфером обмена из терминала, то есть заменяет xclip и xsel. Особенно полезно для пользователей vim, в котором начинает работать копирование/вставка по `"+y` / `"+p` (просто установите этот пакет, дополнительное конфигурирование не требуется).

ydotool — замена xdotool.

waypipe — прокси для сетевой прозрачности. Если честно, я не помню, когда в последний раз кто-нибудь задавал на профильных форумах вопрос, связанный с этой фичей. Может, ей вообще не пользуются? Я — точно нет.

KDE Connect. Добавьте в автозапуск
файл `org.kde.kdeconnect.nonplasma.desktop` (см. предыдущие секции), установите KDE Connect на смартфоне и настройте их связку — и вы сможете управлять со смартфона воспроизведением музыки и видео в браузере с плагином plasma-integration, в vlc, в cantata, в mpv с плагином mpv-mpris, а также получать уведомления в mako. Моё уважение проекту KDE за разработку программ, не прибитых гвоздями к их экосистеме.

Заключение

Писать о настройке всего этого (особенно панели i3blocks) можно бесконечно, однако тут лучше остановиться. Вот что получилось у меня:

Также зайдите на сабреддит /r/unixporn и подивитесь, что делают люди. (впрочем, 95% выложенного там красиво выглядит, но категорически не подходит для повседневной работы :)).