

---

# **Arch Linux Optimization Guide (RU)**

***Выпуск 2022.02.28***

**Arch Linux Optimization Guide (RU)**

**мая 15, 2022**

---

## Содержание:

---

<b>1 Предисловие</b>	<b>1</b>
1.1 Основные консольные команды . . . . .	1
1.1.1 Пару слов об AUR помощниках . . . . .	2
<b>2 Первые шаги</b>	<b>3</b>
2.1 Настройка <code>pacman</code> . . . . .	3
2.1.1 Обновление ключей Arch Linux . . . . .	3
2.1.2 Включение 32-битного репозитория . . . . .	3
2.1.3 Ускорение обновления системы . . . . .	4
2.2 Установка базовых пакетов и набора программ . . . . .	4
2.2.1 Установка актуальных драйверов для видеокарты . . . . .	5
2.2.1.1 NVIDIA . . . . .	5
2.2.1.2 Nouveau ( <i>Только для старых видеокарт</i> ) . . . . .	6
2.2.1.3 AMD . . . . .	6
2.2.1.4 Intel . . . . .	6
2.3 Добавление важных модулей в образы ядра . . . . .	6
2.4 Установка микрокода . . . . .	7
2.5 Настройка драйвера NVIDIA . . . . .	7
2.5.1 Твики драйвера NVIDIA . . . . .	11
2.5.2 Специфические переменные окружения для драйвера NVIDIA . . . . .	12
2.6 Разгон монитора ( <i>Для опытных пользователей</i> ) . . . . .	12
2.6.1 Для видеокарт AMD/Intel . . . . .	12
2.6.2 Для видеокарт NVIDIA . . . . .	14
<b>3 Базовое ускорение системы</b>	<b>19</b>
3.1 Настройка <code>makepkg.conf</code> . . . . .	19
3.1.1 Форсирование использования Clang при сборке пакетов . . . . .	20
3.1.1.1 Ускорение работы компиляторов LLVM/Clang . . . . .	21
3.1.2 Включение <code>ccache</code> . . . . .	22
3.2 Установка полезных служб и демонов . . . . .	22
3.3 Низкие задержки звука . . . . .	24
3.3.1 Новая альтернатива PulseAudio . . . . .	24
3.3.2 Простая ALSA . . . . .	24
3.4 Ускорение загрузки системы (Отключение NetworkManager-wait-online) . . . . .	25
3.4.1 Ускорение загрузки ядра на HDD накопителях ( <i>Только для жестких дисков</i> ) . . . . .	25
3.5 Одновременная загрузка двух и более пакетов . . . . .	26

3.5.1 Альтернативно можно использовать powerpill (Спасибо Zee Captain) . . . . .	26
3.6 Твики драйверов Mesa . . . . .	26
3.6.1 Форсирование использования AMD SAM ( <i>Только для опытных пользователей</i> ) . . . . .	26
3.6.2 Решение проблем работы графики Vega 11 (Спасибо @Vochatrak-az-ezm) . . . . .	27
<b>4 Экстра оптимизации</b>	<b>29</b>
4.1 Перевод процессора из стандартного энергосбережения в режим производительности . . . . .	29
4.1.1 GUI для изменение частоты процессора ( <i>Может не работать с Xanmod</i> ) . . . . .	31
4.1.2 Альтернатива - Auto-Cpufreq . . . . .	32
4.2 Отключение спящего режима и гибернации . . . . .	32
4.3 Отключение дампов ядра ( <i>Только для опытных пользователей</i> ) . . . . .	32
4.4 Отключение файла подкачки . . . . .	33
<b>5 Настройка параметров ядра</b>	<b>34</b>
5.1 Обновление загрузчика и отключение ненужных заплаток . . . . .	34
5.1.1 Разъяснения . . . . .	35
<b>6 Файловые системы</b>	<b>36</b>
6.1 Нюансы выбора файловой системы и флагов монтирования . . . . .	36
6.1.1 Оптимальные флаги монтирования . . . . .	37
6.2 Сжатие в файловой системе Btrfs . . . . .	37
6.2.1 Определение эффективности сжатия . . . . .	39
6.2.2 Скорость обработки алгоритма zstd на примере AMD Ryzen 7 3700X .	39
6.2.3 Список протестированных игр на эффективность сжатия (Спасибо @dewdpol!) . . . . .	41
6.2.3.1 Промежуточные результаты . . . . .	48
<b>7 Кастомные ядра</b>	<b>49</b>
7.1 Выбор ядра . . . . .	49
7.1.1 linux-zen . . . . .	50
7.1.2 liquorix . . . . .	50
7.1.3 Xanmod . . . . .	53
7.1.4 linux-tkg . . . . .	54
7.2 Настройка ядра . . . . .	56
7.3 Сборка ядра с помощью Clang + LTO . . . . .	73
<b>8 Wine / Linux Gaming</b>	<b>77</b>
8.1 Основные составляющие . . . . .	77
8.1.1 Что такое Wine? . . . . .	77
8.1.2 Сборки Wine . . . . .	78
8.1.2.1 Установка wine-staging вместе с зависимостями . . . . .	78
8.1.3 Альтернативные сборки Wine . . . . .	78
8.1.3.1 WINE-TKG . . . . .	79
8.1.3.2 wine-tkg-userpatches . . . . .	80
8.1.3.3 Proton-GE-Custom . . . . .	81
8.1.4 Использование Wine . . . . .	81
8.1.5 DXVK . . . . .	83
8.1.6 vkd3d . . . . .	84
8.1.7 Полезные ссылки по теме Wine и DXVK . . . . .	86
8.2 Дополнительные компоненты . . . . .	86
8.2.1 Lutris . . . . .	86

8.2.1.1 Использование Proton-GE-Custom в Lutris . . . . .	93
8.2.2 Gamemode . . . . .	94
8.2.3 AMD FidelityFX Super Resolution в Wine . . . . .	94
8.2.4 Использование DLSS с видеокартами NVIDIA через Proton . . . . .	97
8.2.5 Мониторинг FPS в играх. . . . .	98
8.2.5.1 Mangohud . . . . .	98
8.2.5.2 Альтернатива: DXVK Hud ( <i>Только для игр запускаемых через Wine/Proton</i> ) . . . . .	99
<b>9 Сборка мини-ядра, и с чем это едят.</b>	<b>100</b>
9.1 Возможные часто встречающиеся проблемы после установки мини-ядра . . . . .	101
<b>10 Оптимизация рабочего окружения (DE)</b>	<b>103</b>
10.1 Запуск любой DE или WM без экранного менеджера ( <i>Только для X11</i> ) . . . . .	103
10.2 GNOME 3.XX/42 . . . . .	104
10.2.1 Удаление мусора GNOME . . . . .	104
10.2.2 Отключение Tracker 3 в GNOME (НОВЫЙ СПОСОБ) . . . . .	104
10.2.3 Отключение ненужных GSD служб GNOME (НОВЫЙ СПОСОБ) . . . . .	105
10.2.4 gnome-shell-performance и mutter-performance . . . . .	106
10.2.5 Бонус: немного косметики . . . . .	107
10.2.6 Результат . . . . .	107
10.3 KDE Plasma 5 . . . . .	108
10.3.1 Удаление мусора из Plasma 5 . . . . .	109
10.3.2 Отключение Baloo в Plasma . . . . .	109
10.3.3 Отключение отладочной информации в KDE 5 . . . . .	110
10.3.4 Отключение ненужных служб Plasma . . . . .	111
10.3.5 Настройка работы KWin для увеличения плавности . . . . .	113
10.3.6 Отключение композитинга для полноэкраных окон . . . . .	115
10.3.7 Отключение ненужных графических эффектов Plasma . . . . .	116
10.3.8 Результат . . . . .	117
10.4 Xfce4 . . . . .	118
10.4.1 Удаление потенциально ненужных компонентов Xfce . . . . .	118
10.4.2 Отключение ненужных служб и приложений автозапуска . . . . .	119
10.4.3 Настройка композитора Xfwm4 . . . . .	119
10.4.4 Результат . . . . .	121
10.5 Cinnamon . . . . .	121
10.5.1 Отключение ненужных CSD служб (НОВЫЙ СПОСОБ) . . . . .	121
10.5.2 Настройка композитора Muffin . . . . .	123
10.5.3 Отключение ненужных эффектов Muffin . . . . .	125
10.5.4 Результат . . . . .	126
<b>11 Полезные программы</b>	<b>127</b>
11.1 Stacer . . . . .	127
11.2 Bleachbit . . . . .	128
11.3 Piper . . . . .	129
11.4 pam_usb . . . . .	130
11.5 Bottles . . . . .	131
<b>Алфавитный указатель</b>	<b>134</b>

# ГЛАВА 1

---

## Предисловие

---

Привет, неизвестный мне чувак из интернета, раз ты тут, то возможно жаждешь настроить свою систему на максимальный выхлоп, но прежде чем начать - знай: **Все манипуляции на вашей совести и авторы не несут никакой ответственности**, но если вам нужна помощь или что-то не понятно - создайте тему на [GitHub Issues](#)

На текущий момент руководство находится в активной переработке, поэтому если вы нашли ошибку или просто хотите внести свой вклад в проект - отправьте на рассмотрение Pull Request в наш GitHub репозиторий. Мы будем признательны за ваше участие.

### 1.1 Основные консольные команды

```
sudo pacman -S      # Установить программу из основных репозиториев.  
sudo pacman -Syu    # Выполнить полное обновление системы и репозиториев  
sudo pacman -R      # Удалить пакет  
sudo pacman -Rsn    # Удалить пакет и его зависимости  
git clone          # Клонирует внешний git репозиторий, например AUR пакет  
makepkg -si         # Осуществляет сборку пакета и его установку из PKGBUILD  
cd                 # Перейти в директорию, например: cd tools.  
ls                 # Показать файлы и папку внутри другой папки.
```

### **1.1.1 Пару слов об AUR помощниках**

Далее в руководстве все пакеты из [AUR](#) (Arch Linux Repository) будут устанавливаться и собираться, если так можно выразиться, "дедовским" способом, т.е. через стандартные утилиты `git` и `makepkg`, без применения так называемых "[AUR Помощников](#)". Это сделано по причине их быстрой сменяемости, и тот помощник который был актуален раньше, может стать устаревшим и никому не нужным. Для примера, так было с AUR-помощником `yaourt`. И кроме того, согласно Arch Wiki AUR-помощники "официально" не поддерживаются дистрибутивом. А "старый" метод, через обычное клонирование `git` репозитория из AUR командой `git clone` и сборка пакета через `makepkg`, будет работать всегда. Тем не менее, обращаем ваше внимание, что возможность установки пакетов через AUR помощник возможна, и вы можете её использовать для всех AUR пакетов о которых пойдет речь далее. Подробнее об этом можно почитать [здесь](#).

# ГЛАВА 2

---

## Первые шаги

---

### 2.1 Настройка pacman

#### 2.1.1 Обновление ключей Arch Linux

Обновление ключей необходимо во избежание дальнейших проблем с установкой пакетов:

```
sudo pacman-key --init          # Инициализация
sudo pacman-key --populate archlinux # Получить ключи из репозитория
sudo pacman-key --refresh-keys    # Проверить текущие ключи на актуальность
sudo pacman -Sy                  # Обновить ключи для всей системы
```

Данная операция может занять продолжительное время.

#### 2.1.2 Включение 32-битного репозитория

Убедимся, что конфигурация пакетного менеджера Pacman настроена для получения доступа к 32-битным зависимостям, нужным в частности для установки Wine и Steam.

Для этого раскомментируем так называемый *multilib* репозиторий:

```
sudo nano /etc/pacman.conf      # Раскоментируйте последние две строчки как на ↵ скриншоте
```

```
[core]
Include = /etc/pacman.d/mirrorlist

[extra]
Include = /etc/pacman.d/mirrorlist

#[community-testing]
#Include = /etc/pacman.d/mirrorlist

[community]
Include = /etc/pacman.d/mirrorlist

# If you want to run 32 bit applications on your x86_64 system,
[multilib]
Include = /etc/pacman.d/mirrorlist
/etc/pacman.conf
```

92,3

95%

```
sudo pacman -Suy # Обновление репозиториев и всех программ
→(пакетов)
```

## 2.1.3 Ускорение обновления системы

Утилита Reflector отсортирует доступные репозитории по скорости:

```
sudo pacman -S reflector rsync curl # Установка reflector и его зависимостей
```

Если вы из Европейской части России, то всегда советуем использовать зеркала Германии, так как их больше всего и они имеют оптимальную свежесть/скорость:

```
sudo reflector --verbose --country 'Germany' -l 25 --sort rate --save /etc/pacman.d/
→mirrorlist
```

Если вы проживаете не на территории Европейской части РФ или в иной стране, то просто измените *Germany* на *Russia* или ваше государство.

Можно также вручную отредактировать список зеркал, добавив туда зеркала из постоянно обновляющегося перечня на сайте Arch Linux (<https://archlinux.org/mirrorlist/>)

```
sudo nano /etc/pacman.d/mirrorlist # Рекомендуем прописывать зеркала от Яндекса
```

## 2.2 Установка базовых пакетов и набора программ

Вот основной набор программ который мы можем вам порекомендовать к установке первым делом:

```
sudo pacman -Syu base-devel \
# Обязательная к установке группа!
nano \
# Минималистичный консольный редактор
git \
# Консольный Git, нужен для установки AUR пакетов
chromium \
# Браузер, на ваш выбор
gvfs \
# Нужен для корзины
ccache \
# Ускоряет дальнейшее перекомпилирование больших пакетов
```

(continues on next page)

(продолжение с предыдущей страницы)

```
vlc \ # Плеер, на ваш выбор
steam \ # Steam, можно также установить steam-runtime
bleachbit \ # Программа очистки системы
grub-customizer \ # Графический менеджер настройки GRUB
unrar \ # Для поддержки архивов формата rar
unzip \ # Для поддержки архивов формата zip
file-roller \ # Минималистичный менеджер архивов
qbittorrent \ # Торрент-клиент, на ваш выбор
unace \ # Для поддержки архивов формата ace
lrzip \ # Для поддержки сжатия через rzip/lzma/lzo и т.д.
squashfs-tools
```

Дополнительно можно отметить легковесный файловый менеджер PCManFM:

```
sudo pacman -S pcmanfm-gtk3 gvfs
```

Мы установили набор джентльмена и парочку программ, что понадобятся нам в дальнейшем. Но если вас не устраивает тот или иной компонент, вы всегда можете найти любой пакет по адресу <https://www.archlinux.org/packages/> или установить из AUR (т.е. скомпилировать) по адресу <https://aur.archlinux.org/packages/>.

## 2.2.1 Установка актуальных драйверов для видеокарты

В установке драйверов для Linux-систем нет ничего сложного, главное просто учитывать, что от свежести ядра и версии драйвера, будет зависеть получите ли вы чёрный экран смерти или нет (Шутка).

**И да, устанавливайте драйвера ТОЛЬКО через пакетный менеджер вашего дистрибутива!**

Забудьте про скачивание драйвера с сайта NVIDIA/AMD, это поможет вам избежать кучу проблем в дальнейшем.

### 2.2.1.1 NVIDIA

В гайде мы установим драйвер версии DKMS, который сам подстроится под нужное ядро и не позволит убить систему при обновлении (не касается свободных драйверов Mesa).

Перед установкой рекомендуется отключить "Secure Boot" в UEFI, ибо из-за этого модули драйвера могут не загрузиться.

```
sudo pacman -S nvidia-dkms nvidia-utils lib32-nvidia-utils nvidia-settings vulkan-icd-loader lib32-vulkan-icd-loader lib32-opencl-nvidia opencl-nvidia libxnvctrl
sudo mkinitcpio -P # Обновляем образы ядра
```

### 2.2.1.2 Nouveau (Только для старых видеокарт)

Для старых видеокарт Nvidia (ниже GeForce 600) рекомендуется использовать свободную альтернативу драйвера NVIDIA — Nouveau, входящую в состав Mesa. Она имеет официальную поддержку и обновления в отличии от старых версий закрытого драйвера NVIDIA (340, 390) и отлично справляется с 2D ускорением. Вдобавок, Nouveau хорошо работает с Wayland.

```
sudo pacman -S mesa lib32-mesa xf86-video-nouveau vulkan-icd-loader lib32-vulkan-icd-  
→ loader
```

### 2.2.1.3 AMD

```
sudo pacman -S mesa lib32-mesa vulkan-radeon lib32-vulkan-radeon vulkan-icd-loader  
→ lib32-vulkan-icd-loader
```

### 2.2.1.4 Intel

```
sudo pacman -S mesa lib32-mesa vulkan-intel lib32-vulkan-intel vulkan-icd-loader  
→ lib32-vulkan-icd-loader
```

Данные команды выполняют установку полного набора драйверов для вашей видеокарты и всех зависимостей, но внимание: автор использует проприетарный драйвер NVIDIA, поэтому если вы заметили ошибку или желаете более проверенный источник: [GitHub](#).

**Внимание:** У авторов отсутствует оборудование AMD, поэтому в данном руководстве основной акцент будет сделан именно на настройке оборудования от компании NVIDIA. Если у вас есть желание дополнить это руководство специфичными для открытых драйверов Mesa твиками/оптимизациями, вы можете отправить нам свои изменения в качестве Pull Request'a на рассмотрение.

## 2.3 Добавление важных модулей в образы ядра

Прежде чем мы начнем, необходимо добавить важные модули в загрузочный образ нашего ядра. Это позволит нам избежать проблем в дальнейшем, и снизить риск словить "чёрный экран" при загрузке из-за того что какие-либо модули не были подгружены во время или просто отсутствуют.

Для этого отредактируем параметры сборки наших образов: `sudo nano /etc/mkinitcpio.conf`

Отредактируйте строку `MODULES` как показано на изображении и выполните команды ниже.

В массив (ограничен скобками) вы можете прописать любые модули ядра которые считаете наиболее важными и нужными. Ниже мы указали модули файловой системы Btrfs.

Если у вас видеокарта от AMD/Intel, то можно прописать дополнительно указать модули соответствующих драйверов AMD/Intel: `amdgpu radeon` или `crc32c-intel intel_agp i915`.

Так же если у вас другая файловая система, то прописывать модули для Btrfs не нужно.

```
MODULES=(crc32c libcrc32c zlib_deflate btrfs)
```

```
GNU nano 5.3                               /etc/mkinitcpio.conf
# vim:set ft=sh
# MODULES
# The following modules are loaded before any boot hooks are
# run. Advanced users may wish to specify all system modules
# in this array. For instance:
#     MODULES=(piix ide_disk reiserfs)
MODULES=(nvidia nvidia_modeset nvidia_uvm nvidia_drm)
```

```
sudo mkinitcpio -P                                # Пересобираем наши образы ядра.
```

## 2.4 Установка микрока

МикроКод - программа реализующая набор инструкций процессора. Она уже встроена в материнскую плату вашего компьютера, но скорее всего вы его либо не обновляли вовсе, либо делаете это не часто вместе с обновлением BIOS (UEFI).

Однако у ядра Linux есть возможность применять его обновления прямо во время загрузки. Обновления микрока содержат множественные исправления ошибок и улучшения стабильности, поэтому настоятельно рекомендуется их периодически устанавливать.

Осуществляется это следующими командами:

```
sudo pacman -S intel-ucode          # Установить микроКод Intel
sudo pacman -S amd-ucode            # Установить микроКод AMD
sudo mkinitcpio -P                 # Пересобираем образы ядра.
sudo grub-mkconfig -o /boot/grub/grub.cfg    # Обновляем загрузчик, можно так же через grub-customizer.
```

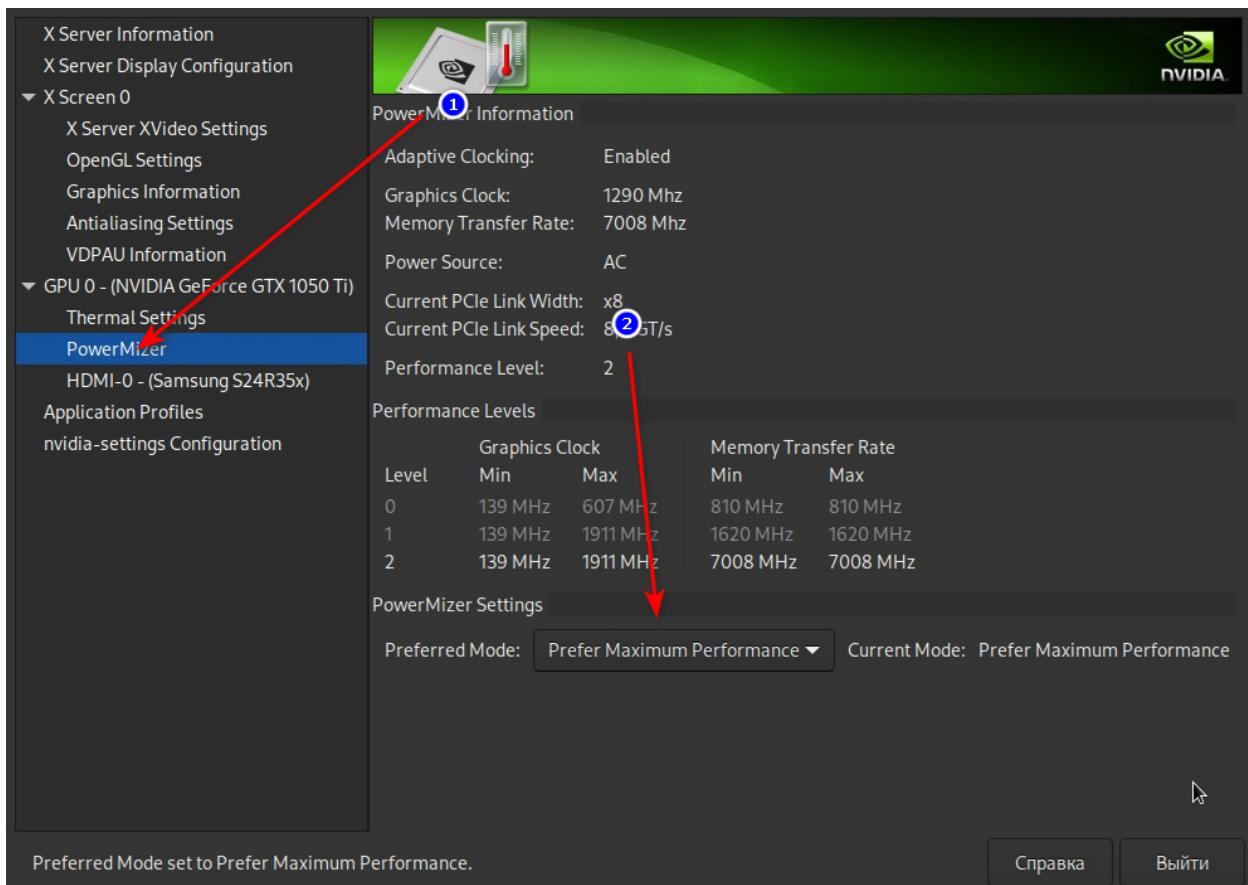
## 2.5 Настройка драйвера NVIDIA

После установки драйвера обязательно перезагрузитесь, откройте панель nvidia-settings, и выполните все шаги как показано на изображениях:

```
nvidia-settings # Открыть панель Nvidia
```



(Если у вас больше одного монитора, то выбирайте здесь тот, который имеет большую частоту обновления)



(Это изменение профиля питания видеокарты работает только до перезагрузки. Если вы хотите зафиксировать профиль производительности, то установите пакет nvidia-tweaks с параметром `_powermizer_scheme=1`, как описано в следующем подразделе.)



(Не забудьте здесь настроить все мониторы которые у вас есть, задать им правильное разрешение и частоту обновления.)

**Внимание:** Советуем вам не использовать параметры "Force composition Pipeline" и "Force Full composition Pipeline". Несмотря на то, что эти два параметра действительно могут полностью вылечить тиринг (разрывы экрана), они также создают сильные задержки ввода (input lag). Вместо этого рекомендуем вам выполнить настройку композитора вашего DE (WM) как это описано в разделе "[Оптимизация рабочего окружения \(DE\)](#)".



Теперь переместите ранее сохраненный файл настройки в `/etc/X11/xorg.conf`, чтобы примененные вами настройки для мониторов работали для всей системы и не слетали после перезагрузки:

```
sudo mv ~/xorg.conf /etc/X11/xorg.conf
```

**Внимание:** Если вы используете GNOME/Plasma, то помните, что эти окружения могут игнорировать настройки для мониторов которые вы указали здесь, и использовать свои собственные. В этом случае настраивать мониторы нужно именно в настройках вашего рабочего окружения.

## 2.5.1 Твики драйвера NVIDIA

По умолчанию в закрытом NVIDIA драйвере не используются некоторые скрытые оптимизации которые могут помочь с улучшением производительности и работоспособности видеокарты. Например, по умолчанию драйвер работает в режиме совместимости с PCIe 2, поэтому у некоторых пользователей драйвера по умолчанию не задействуется высокоскоростная шина PCIe 3.0.

Поэтому, для того чтобы вы могли их активировать удобным способом, мы сделали пакет который включает в себя все эти твики для драйвера - [nvidia-tweaks](#). Прежде чем устанавливать выполните установку самого драйвера NVIDIA как это было описано выше.

### Установка

```
git clone https://aur.archlinux.org/nvidia-tweaks.git
cd nvidia-tweaks
nano PKGBUILD # В PKGBUILD вы можете найти больше опций для настройки, например ↵
makepkg -sric
```

При возникновении следующей ошибки:

```
--> ОШИБКА: Cannot find the fakeroot binary.
--> ОШИБКА: Cannot find the strip binary required for object file stripping.
```

Выполните: `sudo pacman -S base-devel`

## 2.5.2 Специфические переменные окружения для драйвера NVIDIA

Указать вы их можете либо в Lutris для конкретных игр, либо в "Параметрах Запуска" игры в Steam ("Свойства" -> "Параметры запуска"). После указания всех переменных обязательно добавьте в конце "%command%", для того чтобы Steam понимал, что вы указали именно системные переменные окружения для запуска игры, а не параметры специфичные для этой самой игры).

**\_GL\_THREADED\_OPTIMIZATIONS=1 (По умолчанию выключено)** - Активируем многопоточную обработку OpenGL. Используете выборочно для нативных игр/приложений, ибо иногда может наоборот вызывать регрессию производительности. Некоторые игры и вовсе могут не запускаться с данной переменной (К примеру, некоторые нативно-запускаемые части Metro).

**\_GL\_MaxFramesAllowed=1 (По умолчанию - 2)** - Задает тип буферизации кадров драйвером. Можете указать значение "3" (Тройная буферизация) для большего количества FPS и улучшения производительности в приложениях/играх с VSync. Мы рекомендуем задавать вовсе "1" (т.е. не использовать буферизацию, подавать кадры так как они есть). Это может заметно уменьшить значение FPS в играх, но взамен вы получите лучшие задержки отрисовки и реальный физический отклик, т.к. кадр будет отображаться вам сразу на экран без лишних этапов его обработки.

**\_GL\_YIELD="USLEEP" (По умолчанию без значения)** - Довольно специфичный параметр, "USLEEP" - снижает нагрузку на CPU и некоторым образом помогает в борьбе с тирингом, а "NOTHING" дает больше FPS при этом увеличивая нагрузку на процессор.

## 2.6 Разгон монитора (Для опытных пользователей)

Вопреки мнению многих людей, в Linux таки возможно выполнить разгон монитора. Пусть и с небольшим количеством манипуляций мы попробуем это сделать в данном разделе для разных конфигураций оборудования.

**Предупреждение:** Описанные ниже способы не работают для Wayland сессий.

### 2.6.1 Для видеокарт AMD/Intel

Данный способ работает только для драйверов Mesa и Xorg.

Установим все необходимые компоненты:

```
sudo pacman -S xorg-xrandr libxcvt
```

Для начала сгенерируем модельную линию, которая предоставляет Xorg серверу информацию о подключенном мониторе компьютера. Выполните следующую команду, где сначала указываете желаемое разрешение через пробел, а затем и желаемую частоту обновления:

```
cvt 1920 1080 75
```

Теперь зарегистрируем полученную модельную линию в Xorg через утилиту xrandr. Скопируйте выведенную cvt строку и вставьте все после *Modeline* в эту команду:

```
xrandr --newmode "1920x1080_75.00" 220.75 1920 2064 2264 2608 1080 1083 1088 1130 -
→hsync +vsync
```

Теперь применим полученный Modeline для нужного монитора:

```
xrandr --addmode HDMI-0 1920x1080_75.00
xrandr --output HDMI-0 --mode 1920x1080_75.00
```

(Где *HDMI-0* - тип подключения вашего монитора, его можно узнать через команду xrandr без аргументов)

Теперь вы можете в таком порядке выполнять эти операции постепенно повышая частоту обновления монитора, и результат в виде модельной линии с максимальной рабочей частотой обновления добавить в файл настройки Xorg. Например:

```
sudo nano /etc/X11/xorg.conf.d/10-monitor.conf # Прописываем строчки ниже

Section "Monitor"
    Identifier "VGA1" # Здесь указываем тип подключения вашего монитора
    Modeline "1280x1024_60.00" 109.00 1280 1368 1496 1712 1024 1027 1034 1063 -
→hsync +vsync # Здесь указываем модельную линию которая у вас получилась
    Option "PreferredMode" "1280x1024_60.00" # Здесь заменяем на название полученной
→модельной линии
EndSection

Section "Screen"
    Identifier "Screen0"
    Monitor "VGA1" # Здесь указываем тип подключения вашего монитора
    DefaultDepth 24
    SubSection "Display"
        Modes "1280x1024_60.00" # Здесь меняем на название полученной модельной линии
    EndSubSection
EndSection

Section "Device"
    Identifier "Device0"
    Driver "intel"      # Здесь меняем на драйвер вашей видеокарты
EndSection
```

**Внимание:** Обратите внимание на комментарии в приведенном примере файла настройки!

После перезагрузки все настройки должны работать правильно.

Отдельным случаем стоит рассмотреть разгон матрицы ноутбука с графикой Intel. Об этом вы можете прочитать [в данной статье](#).

## 2.6.2 Для видеокарт NVIDIA

Сейчас мы будем рассматривать вопрос разгона монитора только для видеокарт NVIDIA, т. к. у этого производителя есть некоторые проблемы с применением модельных линий Xorg напрямую через XRandr.

Прежде всего, нужно узнать какой тип подключения у вашего монитора, сделать это можно при помощи утилиты xrandr:

```
sudo pacman -S xorg-xrandr # Установка
xrandr           # Запуск
```

Из информации о наших мониторах, выводимой xrandr, нас интересует:

1. Тип подключения монитора который вы хотите разогнать (HDMI-0/DP-0 и т.д.)
2. Строчка с разрешением монитора для разгона. Необходимо чтобы рядом со значением его частоты обновления был знак звездочки (\*). Это означает, что монитор способен выдавать большее количество Герц чем указано, т.е. его можно разогнать.

Затем переходим в панель управления NVIDIA X Settings:

```
sudo nvidia-settings
```

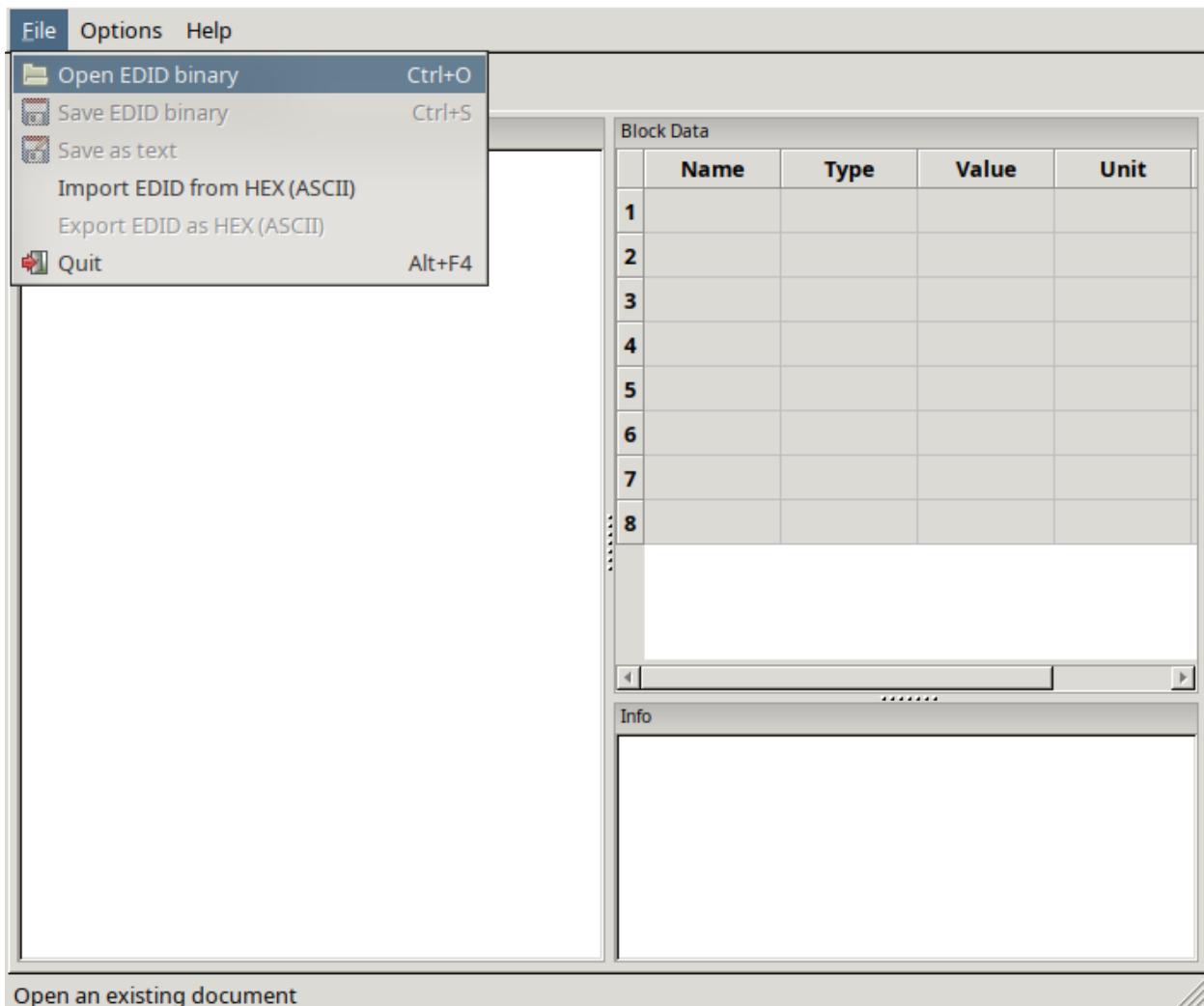
В ней нам нужно полностью настроить наш разгоняемый монитор с соответствующим типом подключения во вкладке "X Server Display Configuration". Задайте разрешение монитора и его частоту обновления согласно тем значениям, что нам вывел xrandr и сохраните все настройки в xorg.conf через кнопку снизу: "Save X Configuration File".

После этого переходим во вкладку с названием монитора который вы хотите разогнать. К примеру: "HDMI-0 - (Samsung S24R35x)". И жмакаем на кнопку "Acquire EDID..." -> И сохраняем EDID файл вашего монитора в домашнюю директорию (Это **обязательный шаг**, сохранять нужно только в домашнюю папку вашего пользователя).

Итак, теперь нам нужно отредактировать наш edid.bin файл монитора. Чтобы это сделать установим свободно распространяемую утилиту wxedid:

```
git clone https://aur.archlinux.org/wxedid.git # Скачивание исходников
cd wxedid                                     # Переход в директорию
makepkg -srcc                                     # Сборка и установка
```

Запустив эту программу откроем через меню наш сохраненный edid файл.

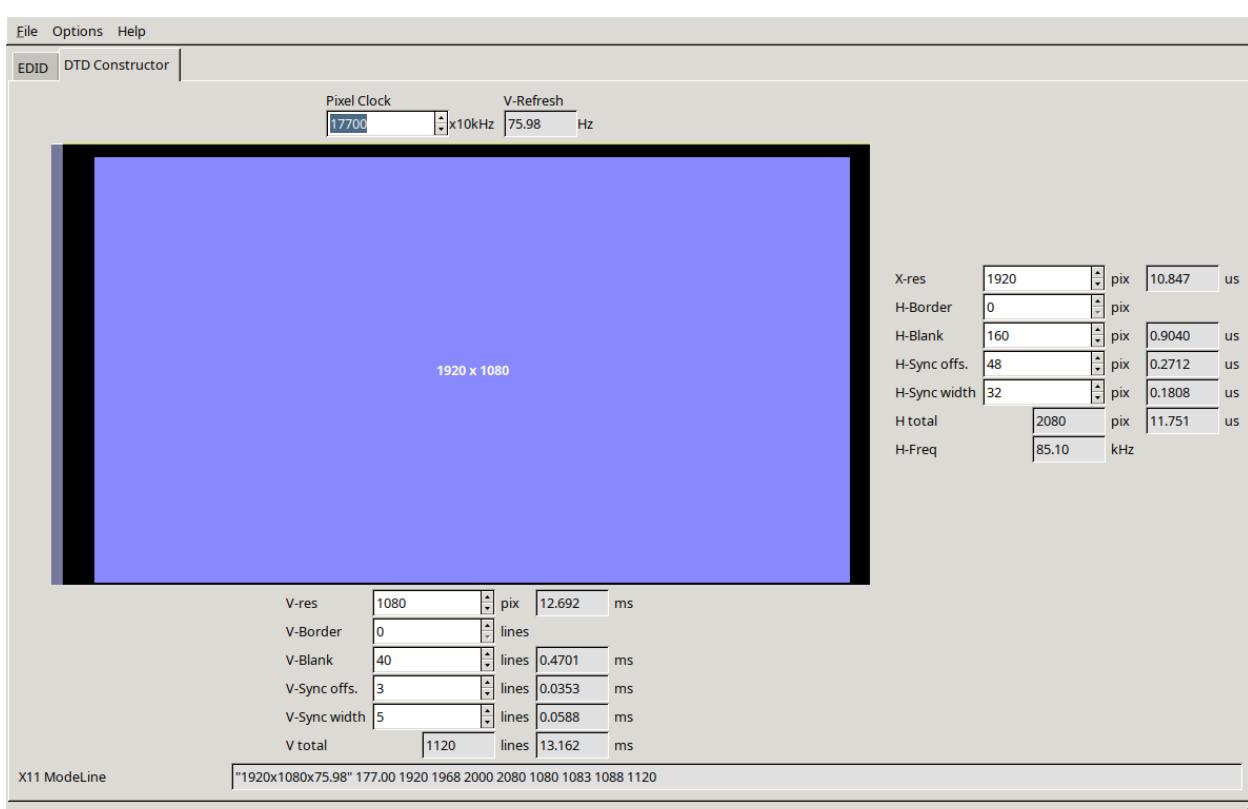


Open an existing document

Затем перейдем в "DTD: Detailed Timing Descriptor".

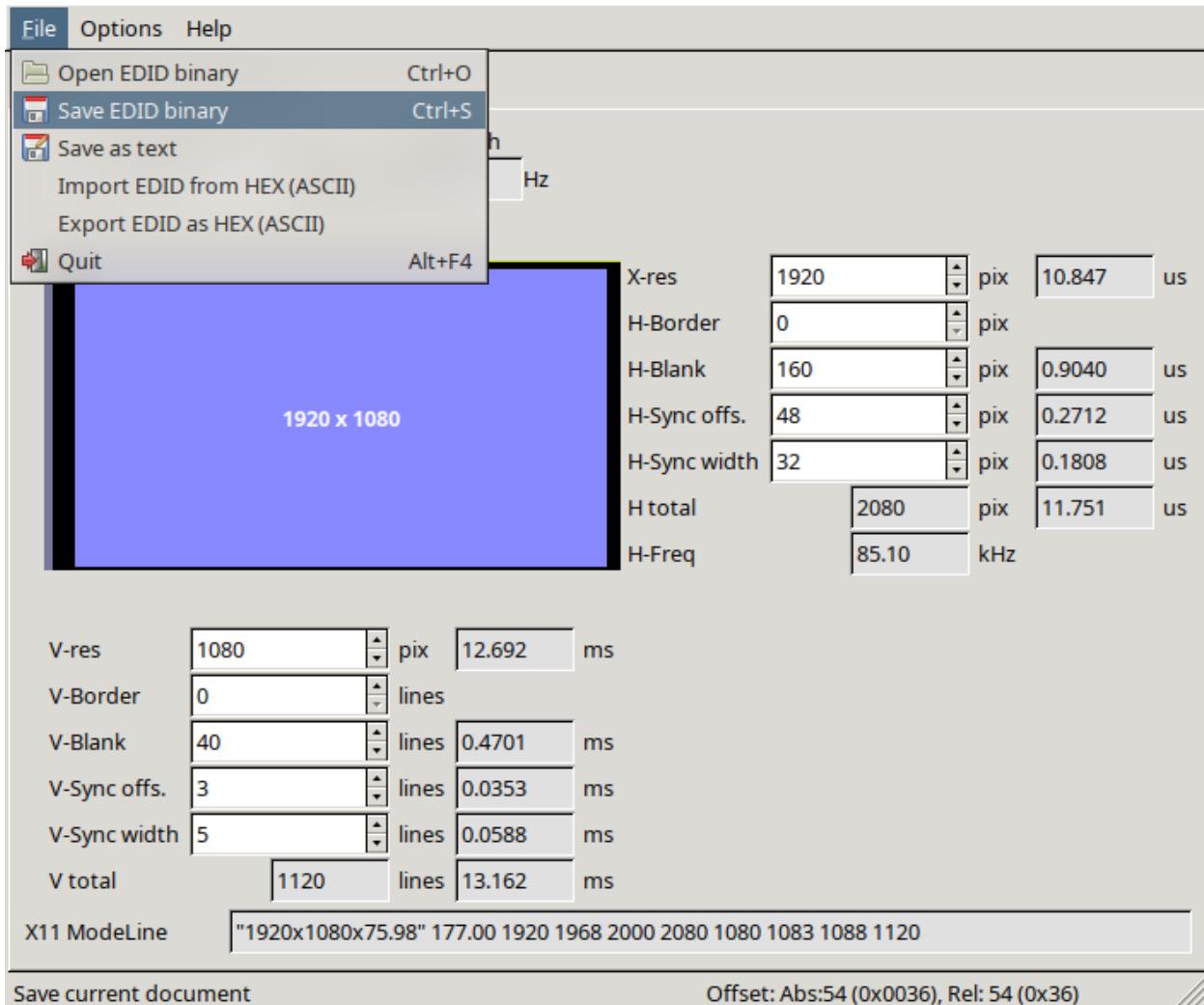


Здесь нужно переключиться на вкладку "DTD Constructor", и в поле "Pixel clock" постепенно повышать частоту обновления монитора до необходимого значения.



О том, как найти нужное значение для вашего монитора - думайте сами и ищите на специализированных ресурсах. Для разных мониторов - разные значения.

Сохраняем уже измененный EDID файл (так же в домашнюю директорию) и закрываем программу.



Теперь в настройках Xorg нужно указать путь до измененного EDID файла в секции с тем монитором который мы разгоняем:

```
sudo nano /etc/X11/xorg.conf # Редактируем ранее сохраненный xorg.conf
```

И добавляем туда опцию с полным путем к измененному EDID файлу в таком формате:

```
Option "CustomEDID" "HDMI-0:/home/ваше_имя_пользователя/edid.bin"
```

(Где *HDMI-0* - ваш тип подключения, а *edid.bin* ваш файл для разгона)

Все. Теперь нужно перезагрузиться и наслаждаться плавностью. (При условии что вы указали правильное значение).

**Предупреждение:** Пользователи с VGA подключением монитора (и не только) могут испытывать проблему с черным экраном после перезагрузки. Поэтому, просим вас заранее сделать себе флешку с записанным на нее любым LiveCD окружением, для того чтобы можно было откатить изменения в случае возникновения проблем.

**Видео версия (Немного устарела)**

<https://www.youtube.com/watch?v=B9o5b2A2qN0>

# ГЛАВА 3

---

## Базовое ускорение системы

---

Переходя к базовой оптимизации системы мне стоит напомнить, что чистый Arch Linux - это фундамент, и требуется уйма надстроек для нормальной работы системы. Установить компоненты, которые будут отвечать за электропитание, чистку, оптимизацию и тому подобные вещи, что и описывается в данном разделе.

### 3.1 Настройка makepkg.conf

Прежде чем приступать к сборке пакетов, мы должны изменить так называемые флаги компиляции, что являются указателями для компилятора, какие инструкции и оптимизации использовать при сборке программ.

`sudo nano /etc/makepkg.conf # Редактируем (Где "-O2" - Это не нуль/ноль)`

**Изменить ваши значения на данные:**

```
CFLAGS="-march=native -mtune=native -O2 -pipe -fno-plt -fexceptions \
-Wp,-D_FORTIFY_SOURCE=2 -Wformat -Werror=format-security \
-fstack-clash-protection -fcf-protection"
CXXFLAGS="$CFLAGS -Wp,-D_GLIBCXX_ASSERTIONS"
RUSTFLAGS="-C opt-level=3"
MAKEFLAGS="-j$(nproc) -l$(nproc)"
OPTIONS=(strip docs !libtool !staticlibs emptydirs zipman purge !debug lto)
```

Данные флаги компилятора выжимают максимум производительности при компиляции, но могут вызывать ошибки сборки в очень редких приложениях. Если такое случится, то отключите параметр 'lto' в строке options добавив перед ним символ восклицательного знака ! ("!lto").

### 3.1.1 Форсирование использования Clang при сборке пакетов

В системах на базе ядра Linux различают две основных группы компиляторов, это LLVM и GCC. И те, и другие хорошо справляются с возложенными на них задачами, но LLVM имеет чуть большее преимущество с точки зрения производительности при меньших потерях в качестве конечного кода. Поэтому, в целом, применение компиляторов LLVM для сборки различных пакетов при задании флага `-O2` (максимальная производительность) является совершенно оправданным, и может дать реальный прирост при работе программ.

Компилятором для языков C/C++ в составе LLVM является Clang и Clang++ соответственно. Его использование при сборке пакетов мы и будем форсировать через `makepkg.conf`

Для начала выполним их установку:

```
sudo pacman -Syu llvm clang lld
```

Теперь клонируем уже готовый конфигурационный файл `/etc/makepkg.conf` под новыми именем `/etc/makepkg-clang.conf`:

```
sudo cp -r /etc/makepkg.conf /etc/makepkg-clang.conf
```

Это поможет нам в случае чего откатиться к использованию компиляторов GCC если возникнут проблемы со сборкой пакетов через LLVM/Clang.

Теперь откроем выше скопированный файл и добавим туда после строки `CHOST="x86_64-pc-linux-gnu"` следующее:

```
export CC=clang
export CXX=clang++
export LD=ld.lld
export CC_LD=lld
export CXX_LD=lld
export AR=llvm-ar
export NM=llvm-nm
export STRIP=llvm-strip
export OBJCOPY=llvm-objcopy
export OBJDUMP=llvm-objdump
export READELF=llvm-readelf
export RANLIB=llvm-ranlib
export HOSTCC=clang
export HOSTCXX=clang++
export HOSTAR=llvm-ar
export HOSTLD=ld.lld
```

Отлично, теперь вы можете собрать нужные вам пакеты (программы) через LLVM/Clang просто добавив к уже известной команде `makepkg` следующие параметры:

```
makepkg --config /etc/makepkg-clang.conf -sric
```

**Внимание:** Далеко не все пакеты так уж гладко собираются через Clang, в частности не пытайтесь собирать им Wine/DXVK, т.к. это официально не поддерживается и с 98% вероятностью приведет к ошибке сборки. Но в случае неудачи вы всегда можете использовать компиляторы GCC, которые у вас заданы в настройках `makepkg.conf` по

умолчанию, т.е. просто уберите опцию `--config /etc/makepkg-clang.conf` из команды `makepkg`.

Дальнейшая пересборка пакетов из официальных репозиториев осуществляется через следующие команды:

```
git clone --depth 1 --branch packages/package https://github.com/archlinux/svntogit-
↪ packages.git package
cd package/trunk
makepkg --config /etc/makepkg-clang.conf -sric --skippgpcheck
```

Где `package` - название нужного вам пакета.

Мы рекомендуем вам пересобрать наиболее важные пакеты. Например такие как драйвера (то есть `mesa`, `lib32-mesa`, если у вас Intel/AMD), `Xorg` сервер, а также связанные с ним компоненты, или `Wayland`, критически важные пакеты вашего DE/WM, например: `gnome-shell`, `plasma-desktop`. А также композиторы `kwin`, `mutter`, `picom` и т.д. в зависимости от того, чем именно вы пользуетесь.

Альтернативно, вы можете использовать уже подготовленный репозиторий `arch-packages` с полной поддержкой сборки пакетов через LLVM/Clang. В этом репозитории представлены не все возможные пакеты, но самые важные компоненты системы там есть, включая сам `llvm-git`, который вы тоже можете пересобрать:

```
git clone https://github.com/h0tc0d3/arch-packages
cd arch-packages
cd llvm-git
makepkg -sric
```

(Вместо `llvm-git` может быть любой другой пакет, доступный в данном репозитории)

Больше подробностей по теме вы можете найти в данной статье:

<https://habr.com/ru/company/ruvds/blog/561286/>

### 3.1.1.1 Ускорение работы компиляторов LLVM/Clang

Дополнительно можно отметить, что после установки Clang вы можете перекомпилировать его самого через себя, т.е. выполнить пересборку Clang с помощью бинарного Clang из репозиториев. Это позволит оптимизировать уже сам компилятор под ваше железо и тем самым ускорить его работу при сборке уже других программ. Аналогичную операцию вы можете проделать и с GCC.

Делается это так же, как и с любыми другими пакетами из официальных репозиториев:

```
git clone --depth 1 --branch packages/clang https://github.com/archlinux/svntogit-
↪ packages.git clang
cd clang/trunk
makepkg --config /etc/makepkg-clang.conf -sric --skippgpcheck
```

### 3.1.2 Включение ccache

В Linux системах есть не так много программ, сборка которых может занять больше двух часов, но они все таки есть. Потому, было бы неплохо ускорить повторную компиляцию таких программ как Wine/Proton-GE и т.д.

ccache - это кэш для компиляторов C/C++, в частности совместимый с компиляторами GCC/Clang, цель которого состоит в ускорении повторного процесса компиляции одного и того же кода. Это значит, что если при сборке программы новой версии, будут замечены полностью идентичные блоки исходного кода в сравнении с его старой версией, то компиляция этих исходных текстов производиться не будет. Вместо этого, уже готовый, скомпилированный код старой версии будет вынут из кэша ccache. За счёт этого и достигается многократное ускорение процесса компиляции.

#### Установка

```
sudo pacman -S ccache
```

После установки его ещё нужно активировать в ваших настройках makepkg. Для этого отредактируем конфигурационный файл:

```
sudo nano /etc/makepkg.conf

# Найдите данную строку в собственных настройках, затем уберите восклицательный знак „!
# перед *"ccache"*
BUILDENV=(!distcc color ccache check !sign)
```

После этого повторная пересборка желаемых программ и их обновление должны значительно ускориться.

**Внимание:** ccache может ломать сборку некоторых программ, поэтому будьте внимательны с его применением.

## 3.2 Установка полезных служб и демонов

**1. Zramswap** — это специальный демон, который сжимает оперативную память ресурсами центрального процессора и создает в ней файл подкачки. Очень ускоряет систему вне зависимости от количества памяти, однако добавляет нагрузку на процессор, т.к. его ресурсами и происходит сжатие памяти. Поэтому, на слабых компьютерах с малым количеством ОЗУ, это может негативно повлиять на производительность в целом.

```
git clone https://aur.archlinux.org/zramswap.git # Скачивание исходников.
cd zramswap                                     # Переход в zramswap.
makepkg -srcc                                     # Сборка и установка.
sudo systemctl enable --now zramswap.service      # Включаем службу.
```

**1.1 Nohang** — это демон повышающий производительность путём обработки и слежки за потреблением памяти.

```
git clone https://aur.archlinux.org/nohang-git.git # Скачивание исходников.
cd nohang-git                                     # Переход в nohang-git
```

(continues on next page)

(продолжение с предыдущей страницы)

```
makepkg -sric                                # Сборка и установка.
sudo systemctl enable --now nohang-desktop      # Включаем службу.
```

**1.2 Ananicy CPP** — это форк одноименного демона, распределяющий приоритет задач. Его установка очень сильно повышает отклик системы. В отличии от оригинального Ananicy, данный форк переписан полностью на C++, из-за чего достигается прирост в скорости работы.

```
git clone https://aur.archlinux.org/ananicy-cpp.git # Скачивание исходников.
cd ananicy-cpp                                     # Переход в ananicy-cpp.
makepkg -sric                                       # Сборка и установка.
sudo systemctl enable --now ananicy-cpp            # Включаем службу.

# Далее описывается установка дополнительных правил по перераспределению приоритетов
# →процессов
git clone https://aur.archlinux.org/ananicy-rules-git.git # Скачивание исходников
cd ananicy-rules-git                               # Переход в директорию
makepkg -sric                                       # Сборка и установка
sudo systemctl restart ananicy-cpp                 # Перезапускаем службу
```

**1.3 Включаем TRIM** — очень полезно для SSD.

```
sudo systemctl enable fstrim.timer               # Включаем службу.
sudo fstrim -v /                                # Ручной метод.
sudo fstrim -va /                               # Если первый метод не тримит весь диск.
```

**1.4 Cron** — это демон, который поможет вам очищать вашу систему от мусора полностью автономно.

```
sudo pacman -S cronie                         # Установить cron.
sudo systemctl enable --now cronie.service      # Запускает и включает службу.
sudo EDITOR=nano crontab -e                   # Редактируем параметр.
```

И прописываем:

```
15 10 * * sun /sbin/pacman -Scc --noconfirm
```

Таким образом наша система будет чистить свой кэш раз в неделю, в воскресенье в 15:10.

**1.5 haveged** - это демон, что следит за энтропией системы. Необходим для ускорения запуска системы при высоких показателях в: *systemd-analyze blame* (Больше 1 секунды).

```
sudo pacman -S haveged                         # Установка
sudo systemctl enable haveged                  # Включает и запускает службу.
```

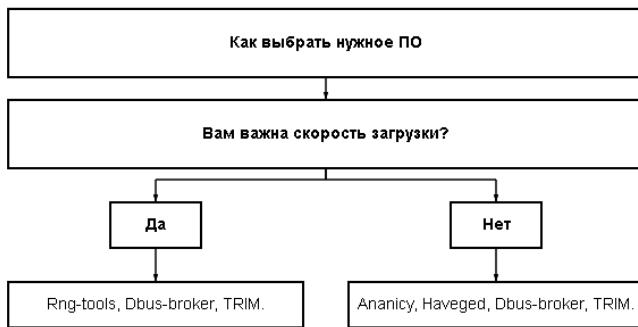
**1.5.1 rng-tools** - демон, что также следит за энтропией системы, но в отличие от haveged уже через аппаратный таймер. Необходим для ускорения запуска системы при высоких показателях *systemd-analyze blame* (Больше 1 секунды).

```
sudo pacman -S rng-tools                      # Установка
sudo systemctl enable rngd                     # Включает и запускает службу.
```

**1.6 dbus-broker** - Это реализация шины сообщений в соответствии со спецификацией D-Bus. Её цель - обеспечить высокую производительность и надежность при сохранении совместимости с эталонной реализацией D-Bus. Обеспечивает чуть более быстрое общение с видеокартой через PCIe.

```
sudo pacman -S dbus-broker          # Установка
sudo systemctl enable --now dbus-broker.service    # Включает и запускает службу.
sudo systemctl --global enable dbus-broker.service # Включает и запускает службу для
                                                    # всех пользователей.
```

Если у вас ещё возникает вопрос: "Что действительно нужно установить из вышеперечисленного?", то просто посмотрите на следующую схему:



### 3.3 Низкие задержки звука

Установите следующие пакеты для понижения задержек звука в PulseAudio, а также удобную графическую панель управления звуком - *pavucontrol*.

```
sudo pacman -S jack2 pulseaudio-alsa pulseaudio-jack pavucontrol jack2-dbus realtime-
                                                    privileges
```

#### 3.3.1 Новая альтернатива PulseAudio

PipeWire - это новая альтернатива PulseAudio, которая призвана избавить от проблем pulse, уменьшить задержки звука и потребление памяти.

```
sudo pacman -S jack2 pipewire pipewire-alsa pavucontrol pipewire-pulse alsu-utils
                                                    wireplumber
```

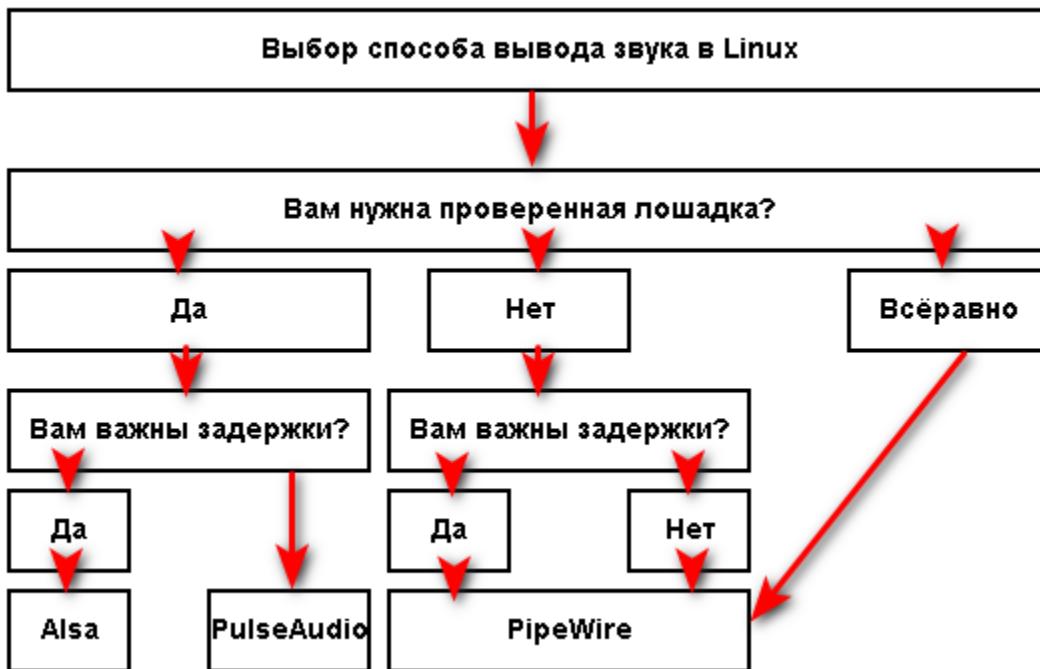
#### 3.3.2 Простая ALSA

ALSA - это тот самый звук (условно, на самом деле это звуковая подсистема ядра), который идёт напрямую из ядра и является самым быстрым, так как не вынужден проходить множество программных прослоек и микширование.

```
sudo pacman -S alsa alsu-utils alsu-firmware alsu-card-profiles alsu-plugins
```

Поэтому, если у вас нет потребности в микшировании каналов, записи аудио через микрофон и вы не слушаете музыку через Bluetooth, то ALSA может вам подойти. Пакет *alsu-utils* также содержит консольный Микшер (настройка громкости), который вызывается командой *alsamixer*.

Вообще, выбор звукового сервера не такая уж сложная задача как вам может показаться, достаточно взглянуть на следующую схему:



## 3.4 Ускорение загрузки системы (Отключение NetworkManager-wait-online)

В большинстве случаев для настройки интернет подключения вы, скорее всего, будете использовать NetworkManager, т.к. он является в этом деле швейцарским ножом и поставляется по умолчанию. Однако, если вы пропишите команду `systemd-analyze blame`, то узнаете, что он задерживает загрузку системы примерно на ~4 секунды. Чтобы это исправить выполните:

```
sudo systemctl mask NetworkManager-wait-online.service
```

### 3.4.1 Ускорение загрузки ядра на HDD накопителях (Только для жестких дисков)

Убедитесь, что пакет `lz4` установлен:

```
sudo pacman -S lz4
```

Отредактируйте файл::

```
sudo nano /etc/mkinitcpio.conf
```

Теперь выполните следующие действия:

- Добавьте `lz4 lz4_compress` в массив `MODULES` (ограничен скобками)
- Раскомментируйте или добавьте строку с надписью `COMPRESSION="lz4"`
- Добавьте строку если её нет - `COMPRESSION_OPTIONS="-9"`
- Добавите `shutdown` в массив `HOOKS` (ограничен скобками)

Это ускорит загрузку системы на слабых жёстких дисках благодаря более подходящему методу сжатия образов ядра.

## 3.5 Одновременная загрузка двух и более пакетов

Начиная с шестой версии pacman поддерживает параллельную загрузку пакетов. Чтобы её включить отредактируйте `/etc/pacman.conf`:

```
sudo nano /etc/pacman.conf # Раскомментируйте строку ниже  
  
# Где 4 - количество пакетов для одновременной загрузки  
ParallelDownloads = 4
```

### 3.5.1 Альтернативно можно использовать powerpill (Спасибо Zee Captain)

```
git clone https://aur.archlinux.org/powerpill.git  
cd powerpill  
makepkg -sric
```

После установки выполните обновление баз данных:

```
sudo powerpill -Syu
```

## 3.6 Твики драйверов Mesa

### 3.6.1 Форсирование использования AMD SAM (Только для опытных пользователей).

AMD Smart Access Memory (или Resizable Bar) — это технология которая позволяет процессору получить доступ сразу ко всей видеопамяти GPU, а не по отдельности для каждого распаянного чипа создавая задержки. Несмотря на то, что данная технология заявлена только для оборудования AMD и требует новейших комплектующих для обеспечения своей работы, мы активируем технологию для видеокарты 10 летней давности ATI Radeon HD 7770 и сравним буст производительности в паре игр.

**Опасно:** Для включения данной технологии в настройках вашего BIOS (UEFI) должна быть включена опция "Re-Size BAR Support" и "Above 4G Decoding". Если таких параметров в вашем BIOS (UEFI) нет - скорее всего технология не поддерживается вашей материнской платой и не стоит даже пытаться её включить.

Чтобы активировать SAM в Linux нужно отредактировать конфигурацию DRI, дописав в конфиг следующие строки:

```
nano ~/.drirc # Редактируем конфигурационный файл

# Прописать строки ниже

<?xml version="1.0" standalone="yes"?>
<driconf>
  <device>
    <application name="Default">
      <option name="radeonsi_enable_sam" value="true" />
    </application>
  </device>
</driconf>
```

Альтернативно её можно активировать через глобальные переменные окружения:

```
sudo nano /etc/environment # Редактируем конфигурационный файл

# Добавить следующие строки
radeonsi_enable_sam=true
# Если используете драйвер RADV
RADV_PERFTEST=sam
```

Проверить работу технологии можно через команду:

```
AMD_DEBUG=info glxinfo | grep smart # Должно быть smart_access_memory = 1
```

### Пример тестирования технологии на видеокарте старого поколения (Windows)

<https://youtu.be/tZmPi9tfLbc>

## 3.6.2 Решение проблем работы графики Vega 11 (Спасибо @Vochatrak-az-ezm)

На оборудовании со встроенным видеоядром Vega 11 может встретиться баг драйвера, при котором возникают случайные зависания графики. Проблема наиболее актуальна для Ryzen 2XXXG и чуть реже встречается на Ryzen серии 3XXXG, но потенциально имеет место быть и на более новых видеоядрах Vega.

Решается через добавление следующих параметров ядра:

```
# Редактируем конфигурационный файл в зависимости от того, какой у вас загрузчик
sudo nano /etc/default/grub

# Параметры можно дописать к уже имеющимся
GRUB_CMDLINE_LINUX_DEFAULT="mdgpu.gttsize=8192 amdgpu.lockup_timeout=1000 amdgpu.gpu_
↪recovery=1 amdgpu.noretry=0 amdgpu.pppfeaturemask=0xffffd3fff amdgpu.deep_color=1_
↪systemd.unified_cgroup_hierarchy=true"
```

На всякий случай можно дописать ещё одну переменную окружения:

```
# Прописать строчку ниже
sudo nano /etc/environment

AMD_DEBUG=nodcc
```

Для подробностей можете ознакомиться со следующими темами:

<https://www.linux.org.ru/forum/linux-hardware/16312119>

<https://www.linux.org.ru/forum/desktop/16257286>

# ГЛАВА 4

## Экстра оптимизации

### 4.1 Перевод процессора из стандартного энергосбережения в режим производительности

По умолчанию ваш процессор динамически меняет свою частоту, что в принципе правильно и дает баланс между энергосбережением и производительностью. Но если вы все таки хотите выжать все соки, то вы можете закрепить применение режима максимальной производительности для вашего процессора. Это также поможет вам избегать "падений" частоты во время игры, которые могли вызывать микрофризы во время игры.

Закрепим режим максимальной производительности:

```
sudo pacman -S cpupower          # Установит менеджер управления  
↪ частотой процессора  
sudo cpupower frequency-set -g performance    # Выставляет максимальную  
↪ производительность до перезагрузки системы.
```

sudo nano /etc/default/cpupower # Редактируем строчку ниже

```
governor='performance'

# Limit frequency range
# Valid suffixes: Hz, kHz (default), MHz, GHz, THz
#min_freq="2.25GHz"
#max_freq="3GHz"

# Specific frequency to be set.
# Requires userspace governor to be available.
# Do not set governor field if you use this one.
#freq=

# Utilizes cores in one processor package/socket fir@@@
/etc/default/cpupower          3,1           11%
```

*governor='performance' # Высокая производительность всегда!*

*sudo systemctl enable cpupower # Включить как постоянную службу которая установит  
вечный perfomance mode.*

### 4.1.1 GUI для изменения частоты процессора (*Может не работать с Xanmod*)



**Установка:**

```
git clone https://aur.archlinux.org/cpupower-gui.git
cd cpupower-gui
makepkg -sric
```

## 4.1.2 Альтернатива - Auto-Cpufreq

Установка:

```
git clone https://aur.archlinux.org/auto-cpufreq-git.git # Скачиваем исходники  
cd auto-cpufreq-git  
makepkg -sric # Переходим в директорию  
systemctl enable auto-cpufreq # Сборка и установка  
# Включает службу как  
systemctl start auto-cpufreq # Запускает службу
```

**Внимание:** Может конфликтовать со встроенным менеджером питания в GNOME 41+. Убедитесь, что он у вас выключен:

```
sudo systemctl disable --now power-profiles-daemon.service
```

## 4.2 Отключение спящего режима и гибернации

sudo pacman -S polkit # Установить для управления системными привилегиями.

sudo nano /etc/polkit-1/rules.d/10-disable-suspend.rules # Убираем спящий режим и гибернацию (из меню и вообще). Если такого файла нет, то создайте его. Файл должен выглядеть вот так:

```
polkit.addRule(function(action, subject) {  
    if (action.id == "org.freedesktop.login1.suspend" ||  
        action.id == "org.freedesktop.login1.suspend-multiple-sessions" ||  
        action.id == "org.freedesktop.login1.hibernate" ||  
        action.id == "org.freedesktop.login1.hibernate-multiple-sessions")  
    {  
        return polkit.Result.NO;  
    }  
});
```

## 4.3 Отключение дампов ядра (**Только для опытных пользователей**)

Отредактируйте */etc/systemd/coredump.conf* в разделе *[Coredump]* раскомментируйте *Storage = external* и замените его на *Storage = none*.

Затем выполните следующую команду:

```
sudo systemctl daemon-reload
```

Уже одно это действие отключает сохранение резервных копий, но они все еще находятся в памяти. Если вы хотите полностью отключить дампы ядра, то измените *soft* на *#\* hard core 0* в */etc/security/limits.conf*.

## 4.4 Отключение файла подкачки

Для лучшей игровой производительности следует использовать файловую систему Btrfs и не задействовать файл подкачки (вместо него стоит использовать выше упомянутый zramswap, конечно при условии что у вас не слишком слабый процессор и оперативной памяти больше чем 4 ГБ), а также без страха отключать фиксы уязвимостей, которые сильно урезают производительность процессора (о них написано в следующем разделе).

```
sudo swapoff /dev/sdxy  # Вместо ху ваше название (Например sdb1).
sudo swapoff -a          # Отключает все swap-разделы/файлы
sudo rm -f /swapfile    # Удалить файл подкачки с диска
sudo nano /etc/fstab     # Уберите самую нижнюю строчку полностью.
```

# ГЛАВА 5

---

## Настройка параметров ядра

---

### 5.1 Обновление загрузчика и отключение ненужных заглаток

По умолчанию в ядре Linux включено довольно много исправлений безопасности, которые однако существенно снижают производительность процессора. Вы можете их отключить через редактирование параметров загрузчика. Рассмотрим на примере GRUB:

```
sudo nano /etc/default/grub # Редактируем настройки вручную или через grub-customizer как на изображении:
```



`sudo grub-mkconfig -o /boot/grub/grub.cfg` # Обновляем загрузчик, можно так же сделать через grub-customizer, добавить и прожать, затем сохранить на 2 и 1 вкладке.

### 5.1.1 Разъяснения

`lpj=` - Уникальный параметр для каждой системы. Его значение автоматически определяется во время загрузки, что довольно трудоемко, поэтому лучше задать вручную. Определить ваше значение для lpj можно через следующую команду: `sudo dmesg | grep "lpj="`

`mitigations=off` - Непосредственно отключает все заплатки безопасности ядра (включая Spectre и Meltdown). Подробнее об этом написано [здесь](#).

`raid=noautodetect` - Отключает проверку на RAID во время загрузки. Если вы его используете - **НЕ** прописывайте данный параметр.

`rootfstype=btrfs` - Здесь указываем название файловой системы в которой у вас отформатирован корень.

`elevator=noop` - Указывает для всех дисков планировщик ввода NONE. **Не использовать если у вас жесткий диск.**

`nowatchdog` - Отключает сторожевые таймеры. Позволяет избавиться от заиканий в онлайн играх.

# ГЛАВА 6

---

## Файловые системы

---

### 6.1 Нюансы выбора файловой системы и флагов монтирования

В отличие от Windows, в Linux-подобных системах выбор файловой системы не ограничивается в обязательном порядке со стороны дистрибутива, и может применяться исходя из личных предпочтений пользователя с оглядкой на поддержку со стороны ядра.

Вот краткое описание основных высокопроизводительных файловых систем:

**EXT4** - универсальный солдат, что подходит, как для SSD носителей, так и для HDD. Имеет самое большое распространение и документацию. Обеспечивает высокие показатели чтения и записи. Из минусов стоит отметить, что данная файловая система начинает проигрывать более новым представителям на рынке и требует ручного допиливания для SSD (SATA, NVMe, PCI и т.п.). Хорошо подходит для домашнего компьютера и файлопомойки, а также серверам которым необходима максимальная стабильность.

**BTRFS** - Т1000 из мира файловых систем. Является наследником идей EXT2-3, и прекрасно подходит для SSD носителей, ибо имеет модули автодетекции, что позволяет не сильно париться с настройкой TRIM и флагов монтирования. Скорость чтения сопоставима, а иногда (Особенно при высоких нагрузках) превышают показатели EXT4. Идеальный выбор для игровой/домашней системы на базе Linux.

### 6.1.1 Оптимальные флаги монтирования

Вот оптимальные параметры для SSD носителей. Описание каждого из них вы можете найти - [здесь](#).

```
rw,relatime,ssd,ssd_spread,space_cache=v2,max_inline=256,commit=600,nodatacow
```

Прежде всего, отметим, что вы можете изменить *relatime* на *noatime* или *lazytime* - все три параметра отвечают за запоминание времени доступа к файлами и прочим связанным с ним атрибутами, что только портит отклик.

Параметр *noatime* полностью выключает данную функцию, что может привести к некоторым багам в приложениях зависимых от времени (например git), но автор никогда не встречал данной проблемы. Параметр *lazytime* успешно будет выполнять все функции времени, но выполнять их промежуточную запись в оперативной памяти, что позволит избежать замедления без потери функционала, но говорят *lazytime* чудит, поэтому автор советует *noatime*.

Но если вы хотите минимум возможных проблем, то оставьте флаг *relatime*.

*space\_cache* можно заменить на *space\_cache=v2*, что тоже даст определенную прибавку производительности.

Прописывать их нужно в файл `/etc/fstab` для корневого и домашнего разделов. Некоторые из данных флагов будут применяться только для новых файлов.

```
GNU nano 6.0
# Static information about the filesystems.
# See fstab(5) for details.

# <file system> <dir> <type> <options> <dump> <pass>
btrfs      rw,noatime,ssd,ssd_spread,space_cache=v2,max_inline=256,nodatacow,subvol=/  0  0
vfat       rw,noatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shortname=mixed.utf8,errors=remount-ro  0  2
```

**Внимание:** Параметр *commit=600* может вызывать повышенное потребление оперативной памяти и портить данные на диске.

**Внимание:** При использовании Btrfs для корневого раздела, обязательно установите пакет `btrfs-progs`.

## 6.2 Сжатие в файловой системе Btrfs

В файловой системе Btrfs есть возможность включения сжатия. Все записываемые файлы по возможности будут сжиматься и экономить пространство на носителе HDD или SSD.

Для SSD это может быть важно в связи с их ограниченным ресурсом на запись.

Согласно [wiki Btrfs](#), официально имеется 3 поддерживаемых алгоритма:

- zlib - высокая степень сжатия, но низкая скорость сжатия и распаковки

- lzo - высокая скорость сжатия и распаковки, но наименьший уровень сжатия из представленных алгоритмов
- zstd - степень сжатия сравнимая с zlib и более быстрые сжатие и распаковка, однако уступающие по скорости lzo

Для включения алгоритма сжатия в файловой системе необходимо:

1. Убедиться в наличии необходимого алгоритма в системе или установить выбранный (zlib, lzo или zstd соответственно).
2. Отредактировать файл `etc/fstab`, добавив для необходимого раздела или носителя следующий флаг монтирования:

```
compress='алгоритм':N
```

(Где N - степень сжатия: для zlib - N = 1,2,...9; для lzo - выбор уровня сжатия не предусмотрен, поэтому :N - не указываются; для zstd - N = 1,2,...15. Чем выше данный параметр, тем сильнее будут сжиматься данные, конечно при условии что это возможно, но также будет повышена нагрузка на процессор, поскольку сжатие выполняется за счет его ресурсов. Согласно [wiki Btrfs](#), оптимальным значением N по отношению степень сжатия / скорость считается 3)

Например для zstd со степенью сжатия 3 запись будет выглядеть примерно следующим образом, если учесть приведенные выше флаги монтирования:

```
rw,relatime,compress=zstd:3,ssd,ssd_spread,space_cache=v2,max_inline=256,commit=600
```

**Внимание:** Сжатие файловой системы не работает вместе с флагом монтирования `nodatacow`.

После выставления данного флага монтирования новые файлы начнут сжиматься при записи на диск. Для сжатия уже имеющихся данных необходимо выполнить команду:

```
sudo btrfs filesystem defragment -c alg /path
```

(Где -c alg - алгоритм (указывается как czlib, clzo или czstd в зависимости от выбранного алгоритма), path - путь к разделу или папке)

Для сжатия уже существующих данных в папке или целого раздела необходимо указать ключ -r перед -c alg:

```
sudo btrfs filesystem defragment -r -c alg /path
```

(Где path - путь к разделу или папке)

**Внимание:** Степень сжатия в данном случае не указывается!

### 6.2.1 Определение эффективности сжатия

Если вы хотите определить эффективность сжатия на вашем разделе/диске, то вам необходимо воспользоваться программой `compsize`. Установить ее можно с помощью команды:

```
sudo pacman -S compsize
```

Для выполнения проверки на эффективность необходимо использовать команду:

```
sudo compsize /path
```

(Где `path` - путь к разделу, папке или файлу)

Пример вывода команды:

```
Processed 309409 files, 1592042 regular extents (1599469 refs), 56369 inline.
Type      Perc     Disk Usage   Uncompressed Referenced
TOTAL     86%      645G        742G        742G
none      100%     619G        619G        619G
zstd      21%       26G         123G        123G
```

Пояснения:

- **Первый столбец:**

- Стока `TOTAL` - итоговые данные, которые учитывают все сжатые и не сжатые файлы и разные алгоритмы (если такие имеются).
- Стока `none` - данные, которые не были сжаты.
- Далее отображаются все использованные алгоритмы (в данном случае - `zstd`).

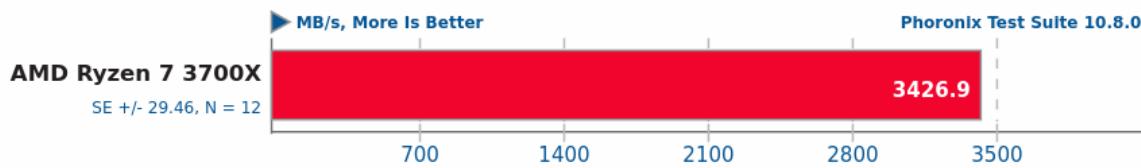
- Второй столбец показывает данные в процентах.
- Третий столбец отображает фактически использованное место на диске/разделе.
- Четвертый столбец показывает данные без сжатия.
- Пятый - видимый размер файла, тот, который зачастую отображается в системе.

### 6.2.2 Скорость обработки алгоритма zstd на примере AMD Ryzen 7 3700X

Для сравнения степеней сжатия алгоритма `zstd` использовалась бенчмарк платформа `phoronix-test-suite`. В данной программе, для проверки скорости сжатия и распаковки данных, доступно три степени - 3, 8, 19. Для получения информации о падении скорости выполнения сжатия нам будет достаточно и первых двух, поскольку степень сжатия 19 на данный момент не поддерживается (однако данные также приведены для ознакомления), и если обратить внимание на полученные данные, то это и не имеет особого смысла. Далее представлены результаты замеров:

### Zstd Compression 1.5.0

Compression Level: 3 - Compression Speed

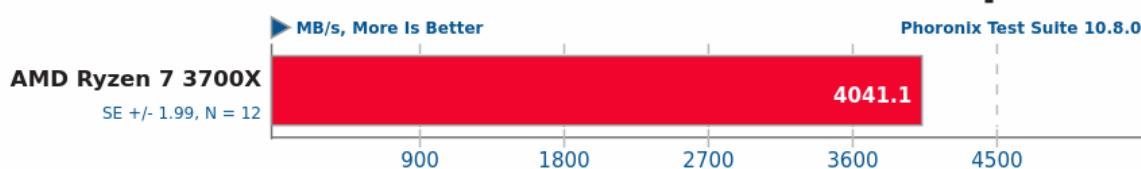


1. (CC) gcc options: -O3 -pthread -lz -llzma -llz4



### Zstd Compression 1.5.0

Compression Level: 3 - Decompression Speed

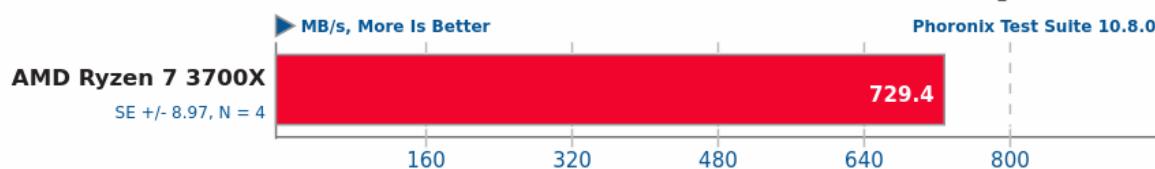


1. (CC) gcc options: -O3 -pthread -lz -llzma -llz4



### Zstd Compression 1.5.0

Compression Level: 8 - Compression Speed

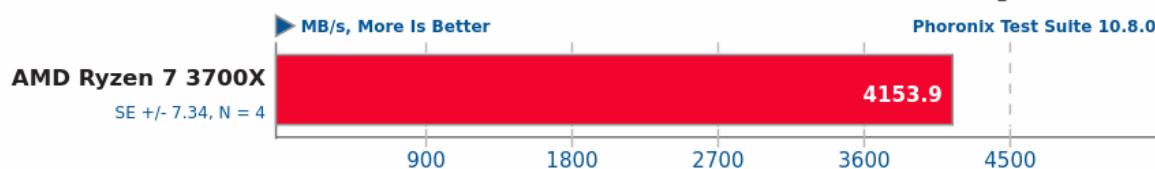


1. (CC) gcc options: -O3 -pthread -lz -llzma -llz4



### Zstd Compression 1.5.0

Compression Level: 8 - Decompression Speed

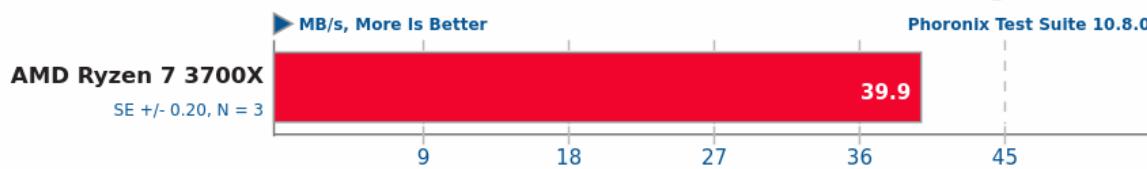


1. (CC) gcc options: -O3 -pthread -lz -llzma -llz4



## Zstd Compression 1.5.0

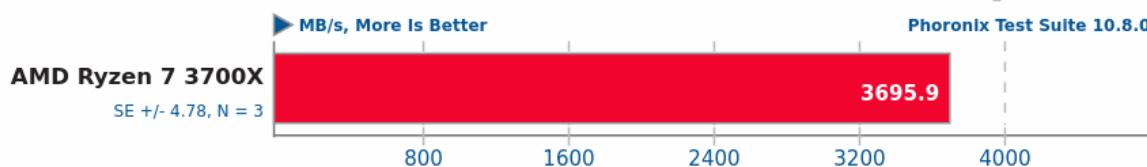
Compression Level: 19 - Compression Speed



1. (CC) gcc options: -O3 -pthread -lz -lzma -lzz4

## Zstd Compression 1.5.0

Compression Level: 19 - Decompression Speed



1. (CC) gcc options: -O3 -pthread -lz -lzma -lzz4

Как можно видеть из графиков, падение скорости при переходе от степени 3 к степени 8 сопровождается падением скорости сжатия более чем в **4,7** раз (не говоря о более высоких степенях сжатия) и практически не изменяется при выполнении распаковки, что может негативно сказаться на скорости установки программ и возможно в некоторых других ситуациях которые требует выполнения записи на диск.

Стоит отметить, что в случае выполнения установки игр с использованием степени сжатия 15, было замечено повышение нагрузки на процессор вплоть до 72-75% в тех случаях, когда файлы поддавались сжатию.

### 6.2.3 Список протестированных игр на эффективность сжатия (Спасибо @dewdpol!)

Далее представлен список протестированных игр на сжатие в файловой системе Btrfs. Данные были получены с помощью программы compsize и являются округленными, поэтому информация может нести частично ознакомительный характер.

№	Игра	Алгоритм	Уровень сжатия	Необходимое место (N)	Используемое место(U)	U/N
1	A Plague Tale: Innocence	zstd	3	41 GB	41 GB	99%
			15			
2	A Story About My Uncle	zstd	3	1,1 GB	1,1 GB	94%
			15			
3	Aegis Defenders	zstd	3	1,3 GB	240 MB	17%

continue

Таблица 1 - продолжение с предыдущей страницы

№	Игра	Алгоритм	Уровень сжатия	Необходимое место (N)	Используемое место(U)	U/
			15		230 MB	16
4	Among Us	zstd	3	429 MB	284 MB	66
			15		279 MB	65
5	Aragami	zstd	3	7,6 GB	5,4 GB	71
			15		5,3 GB	69
6	Armello	zstd	3	1,6 GB	1,5 GB	95
			15			94
7	Bastion	zstd	3	1,1 GB	1,1 GB	94
			15		1,0 GB	93
8	BattleBlockzstd Theater		3	1,8 GB	1,7 GB	93
			15			
9	Beholder	zstd	3	1,9 GB	1,0 GB	55
			15		1,1 GB	58
10	Beholder 2	zstd	3	2,5 GB	2,2 GB	85
			15		2,1 GB	81
11	Blasphemous	zstd	3	854 MB	805 MB	94
			15		802 MB	93
12	Blue Fire	zstd	3	6,0 GB	4,9 GB	81
			15		4,7 GB	77
13	Brothers - A Tale of Two Sons	zstd	3	1,2 GB	1,1 GB	95
			15			
14	Castle Crashers	zstd	3	199 MB	183 MB	92
			15			91
15	Celeste	zstd	3	1,1 GB	897 MB	78
			15		871 MB	75
16	Child of light	zstd	3	2,3 GB	2,3 GB	99
			15			
17	Children of Morta	zstd	3	1,6 GB	1,5 GB	94
			15			
18	CODE VEIN	zstd	3	35 GB	35 GB	99
			15			
19	Cortex Command	zstd	3	97 MB	65 MB	67
			15		64 MB	66
20	Cuphead	zstd	3	3,6 GB	3,3 GB	93
			15			
21	Curse of Dead Gods	zsrd	3	2,7 GB	1,4 GB	53
			15			51
22	D-Corp	zstd	3	1,2 GB	720 MB	57
			15		697 MB	55
23	Dark Souls: Prepare To Die Edition	zstd	3	3,7 GB	3,7 GB	99
			15			
24	Dark Souls III	zstd	3	24 GB	24 GB	99
			15			
25	Darkest Dungeon	zstd	3	3,2 GB	2,8 GB	88

continue

Таблица 1 - продолжение с предыдущей страницы

№	Игра	Алгоритм	Уровень сжатия	Необходимое место (N)	Используемое место(U)	U/N
			15			87
26	Darkestville Cattle	zstd	3	1,7 GB	798 MB	40
			15		682 MB	38
27	Darksiders III	zstd	3	24 GB	24 GB	99
			15			
28	Dead Cells	zstd	3	1,1 GB	1,1 GB	97
			15		1,0 GB	
29	Death's Door	zstd	3	3,6 GB	2,1 GB	58
			15			57
30	Death's Gambit: Afterlife	zstd	3	1 GB	729 MB	66
			15		720 MB	65
31	Deponia: The Complete Journey	zstd	3	9,5 GB	9,5 GB	99
			15			
32	Devil May Cry 5	zstd	3	33 GB	33 GB	99
			15			
33	Disco Elysium	zstd	3	9,5 GB	9,1 GB	96
			15			95
34	Don't Starve Together	zstd	3	2,5 GB	1,8 GB	74
			15			73
35	Eldest Souls	zstd	3	1,0 GB	720 MB	69
			15		708 MB	68
36	Evergate	zstd	3	2,9 GB	1,9 GB	64
			15			63
37	Frostpunk	zstd	3	8,9 GB	8,9 GB	99
			15			
38	Furi	zstd	3	4,3 GB	2,7 GB	62
			15			63
39	Gato Roboto	zstd	3	440 MB	415 MB	94
			15		414 MB	
40	Gears Tactics	zstd	3	29 GB	29 GB	99
			15			
41	Ghost of a Tale	zstd	3	4,7 GB	3,7 GB	79
			15			
42	Ghostrunner	zstd	3	24 GB	20 GB	84
			15			
43	Gibbous - a Cthulhu Adventure	zstd	3	9,0 GB	4,2 GB	47
			15		4,1 GB	46
44	Gris	zstd	3	3,2 GB	1,5 GB	47
			15			46
45	Hades	zstd	3	11 GB	10 GB	95
			15			
46	Hand of Fate	zstd	3	2,5 GB	2,2 GB	90
			15			89
47	Hand of Fate 2	zstd	3	4,1 GB	4,1 GB	99

continue

Таблица 1 - продолжение с предыдущей страницы

№	Игра	Алгоритм	Уровень сжатия	Необходимое место (N)	Используемое место(U)	U/
			15			
48	Hellblade: Sanua's Sacrifice	zstd	3	18 GB	16 GB	87
			15		18 GB	
49	Helldivers	zstd	3	6,4 GB	6,4 GB	99
			15			
50	Hob	zstd	3	2,4 GB	2,2 GB	90
			15		2,1 GB	
51	Hollow Knight	zstd	3	7,5 GB	1,5 GB	20
			15		1,4 GB	
52	Inmost	zstd	3	1,3 GB	649 MB	47
			15		638 MB	
53	Jotun	zstd	3	3,8 GB	1,8 GB	48
			15			
54	Journey	zstd	3	3,3 GB	1,8 GB	55
			15		1,9 GB	
55	Katana ZERO	zstd	3	216 MB	178 MB	82
			15		177 MB	
56	Kate	zstd	3	254 MB	104 MB	40
			15		100 MB	
57	Limbo	zstd	3	98 MB	97 MB	98
			15			
58	Little Nightmare	zstd	3	8,9 GB	5,8 GB	65
			15		4,8 GB	
59	Loop Hero	zstd	3	140 MB	116 MB	83
			15		115 MB	
60	Magicka	zstd	3	1,6 GB	1,6 GB	96
			15			
61	Magicka 2	zstd	3	2,9 GB		99
			15		2,9 GB	
62	Mark of the Ninja: Remastered	zstd	3	7,5 GB	6,9 GB	92
			15			
63	Master of Anima	zstd	3	1,5 GB	1,2 GB	81
			15			
64	METAL GEAR RISING: REVENGEANCE	zstd	3	24 GB	24 GB	99
			15			
65	Moonlight	zstd	3	1,1 GB	577 MB	48
			15		572 MB	
66	Move or Die	zstd	3	666 MB	572 MB	85
			15		567 MB	
67	My Friend Pedro	zstd	3	3,5 GB	2,9 GB	82
			15			
68	Nier:Automata	zstd	3	40 GB	37 GB	91
			15			
69	Nine Parchments	zstd	3	5,7 GB	5,7 GB	98

continue

Таблица 1 - продолжение с предыдущей страницы

№	Игра	Алгоритм	Уровень сжатия	Необходимое место (N)	Используемое место(U)	U/N
			15			
70	Ori and the Blind Forest: Definitive Edition	zstd	3	10 GB	4,9 GB	48%
			15		4,7 GB	
71	Ori and the Will of the Wisps	zstd	3	11 GB	5,5 GB	48%
			15		5,3 GB	
72	Othercide	zstd	3	6,0 GB	5,9 GB	98%
			15			
73	Out of Line	zstd	3	1,3 GB	497 MB	37%
			15		476 MB	
74	Outland	zstd	3	675 MB	593 MB	87%
			15		589 MB	
75	Overcooked 2	zstd	3	7,9 GB	7,7 GB	98%
			15			
76	Papers, Please	zstd	3	58 MB	45 MB	77%
			15		44 MB	
77	Path of Exile	zstd	3	27 GB	27 GB	99%
			15			
78	Peace, Death!	zstd	3	83 MB	76 MB	91%
			15			
79	Peace, Death! 2	zstd	3	34 MB	26 MB	78%
			15			
80	Pummel Party	zstd	3	2,1 GB	1,4 GB	67%
			15			
81	Remember Me	zstd	3	6,7 GB	6,6 GB	99%
			15			
82	Rocket League	zstd	3	18 GB	18 GB	99%
			15			
83	RUINER	zstd	3	10 GB	10 GB	99%
			15			
84	Salt and Sanctuary	zstd	3	563 MB	540 MB	95%
			15			
85	Samorost 1	zstd	3	68 MB	68 MB	99%
			15			
86	Samorost 2	zstd	3	141 MB	141 MB	99%
			15		140 MB	
87	Samorost 3	zstd	3	1,1 GB	1,0 GB	99%
			15			
88	Sekiro: Shadow Die Twice	zstd	3	13 GB	13 GB	99%
			15			
89	Severed Steel	zstd	3	4,0 GB	2,7 GB	68%
			15			
90	Shadow Tactics: Blades of the Shogun	zstd	3	7,3 GB	5,0 GB	69%

continue

Таблица 1 - продолжение с предыдущей страницы

№	Игра	Алгоритм	Уровень сжатия	Необходимое место (N)	Используемое место(U)	U/N
			15		4,8 GB	66
91	Shadowrun Returns	zstd	3	2,8 GB	1,1 GB	39
			15		1,0 GB	37
92	Shattered - Tale of the Forgotten King	zstd	3	6,3 GB	6,3 GB	99
			15			
93	Shiro	zstd	3	80 MB	74 MB	91
			15		73 MB	
94	Skul: The Hero Slayer	zstd	3	1016 MB	1001 MB	98
			15		987 MB	97
95	SpeedRunners	zstd	3	662 MB	651 MB	98
			15		650 MB	
96	Spiritfarer: Farewell	zstd	3	6,0 GB	2,3 GB	38
			15			39
97	Stoneshard Prologue	zstd	3	289 MB	261 MB	90
			15		260 MB	89
98	Stories: The Path of Destinies	zstd	3	1,6 GB	1,6 GB	99
			15			
99	Styx: Master of Shadow	zstd	3	6,7 GB	6,6 GB	98
			15			
100	Styx: Shards of Darkness	zstd	3	10 GB	10 GB	99
			15			
101	Sundered: Eldritch Edition	zstd	3	2,2 GB	1,7 GB	75
			15		1,5 GB	69
102	SYNTHETIK	zstd	3	599 MB	518 MB	86
			15		516 MB	
103	Tabletop Simulator	zstd	3	2,7 GB	1,7 GB	65
			15			63
104	The Escapists 2	zstd	3	2,4 GB	1,7 GB	71
			15			
105	The Life and Suffering of Sir Brante	zstd	3	2,7 GB	1,1 GB	42
			15			43
106	The Cave	zstd	3	1,1 GB	1,1 GB	98
			15			
107	The Red Solstice	zstd	3	2,7 GB	1,4 GB	52

continue

Таблица 1 – продолжение с предыдущей страницы

№	Игра	Алгоритм	Уровень сжатия	Необходимое место (N)	Используемое место(U)	U/N	
			15			51	
108	They Always Run	zstd	3	10 GB	4,2 GB	39	
			15		3,8 GB	34	
109	This War of Mine	zstd	3	2,6 GB	2,5 GB	98	
			15			96	
110	Titan Souls	zstd	3	206 MB	183 MB	88	
			15		182 MB	87	
111	Transistor	zstd	3	3,0 GB	2,7 GB	88	
			15			87	
112	Trine	zstd	3	1,3 GB	1,3 GB	97	
			15			96	
113	Undertale	zstd	3	155 MB	141 MB	90	
			15		140 MB	89	
114	Valiant Hearts: The Great War	zstd	3	1,2 GB	1,1 GB	99	
			15			98	
115	Vanquish	zstd	3	18 GB	18 GB	99	
			15			98	
116	Vesper	zstd	3	2,8 GB	998 NB	34	
			15		964 MB	32	
117	Void Bastards	zstd	3	5,7 GB	2,3 GB	40	
			15			41	
118	Wasteland 2: Director's Cut	zstd	3	14 GB	13 GB	91	
			15			90	
119	Wasteland 3	zstd	3	26 GB	24 GB	91	
			15		23 GB	89	
120	Witch It	zsta	3	4,2 GB	4,1 GB	98	
			15			97	
121	Wizard of Legend	zstd	3	786 MB	475 MB	60	
			15		468 MB	59	
Итого		zstd	3	761 GB	666 GB	87	
			15		664 GB	86	
Кэш шейдеров представляемых здесь игр в Steam		zstd	3	26 GB	25 GB	99	
			15			98	

Примечания:

- По возможности данный список будет расширяться новыми играми и другими алгоритмами сжатия.
- U/N - выраженное в процентах соотношение количества фактически занятого места к необходимому, т.е. если от 100% отнять U/N можно получить процент сэкономленного места на диске. Из чего следует, что чем меньше данный показатель, тем лучше.
- Экономия рассчитывалась вручную с округлением в меньшую сторону. Другими словами, если получалось 1,3087... GB, то записывалось как 1,30 GB.

#### 6.2.3.1 Промежуточные результаты

- **64** игр из представленных **121** - практически не сжимаются, экономия места достигает всего 0-10%.
- **33** игр из представленных **121** - сжимаются с низкой эффективностью, экономия места составляет 11-40%.
- **22** игры из представленных **121** - сжимаются со средней эффективностью, экономия места составляет 41-70%.
- **2** игры из представленных **121** - сжимаются хорошо, экономия места составляет 71-90%.
- Кэш шейдеров, который собирается и хранится на диске в Steam (при включении данной функции) сжимается незначительно - менее 1% экономии.
- С учетом разницы в экономии места порядка **3 GB** между максимальной степенью сжатия 15 и рекомендуемой для Btrfs - 3, и значительного падения скорости выполнения сжатия, можно отметить, что использование степени сжатия выше 3 выглядит крайне сомнительно.

# ГЛАВА 7

---

## Кастомные ядра

---

Прежде чем мы начнем, хотелось бы прояснить такой вопрос: "А зачем вообще нужны эти кастомные ядра?". Чтобы дать ответ на данный вопрос, стоит понимать, что ядро Linux является определенным универсальным стандартом в мире операционных систем, которое одинаково подходит как для домашнего ПК, ноутбука, телефона, так и для сервера, роутера, микро-контроллера. То есть, ядро по умолчанию является швейцарским ножом, позволяющим применять себя в разных задачах, но при этом не быть наилучшим в чём-то конкретном, определенном. Нет, это не значит что на ванильном ядре вы не сможете запустить какую-то игру скажем через Wine или Proton, но такой опыт не будет лучшим, т.к. за такую многопрофильность ядру приходится платить меньшей производительностью в этих самых определенных задачах. Кастомные же ядро подразумевает определенную заточенность под что-то конкретное, делая упор на что-то одно. В нашем случае это производительность и игры, а также улучшение опыта использования Linux на домашнем ПК или ноутбуке. В этом нам и помогут нижеперечисленные ядра вместе с их правильной настройкой.

И ёщё просим вас заранее установить стабильное linux-lts ядро. В случае возникновения проблем вы всегда сможете откатиться на эту версию ядра.

Проверка ядра используемого в данный момент осуществляется следующей командой: `uname -r`.

### 7.1 Выбор ядра

Хотелось бы отметить, что универсального рецепта по сборке "лучшего ядра" не существует, и каждый выбирает то, что конкретно для него лучше работает. Поэтому мы рекомендуем вам установить и попробовать каждое ядро из предложенных ниже на своем железе, проводя тесты в любимых играх/востребованных задачах.

### **7.1.1 linux-zen**

Zen ядро - это плод коллективных усилий по объединению патчей улучшающих опыт домашнего использования Linux, но при этом работающих стабильно и не ломающих совместимость с теми или иными вещами.

Это отличный выбор для неискушенного пользователя, что не ставит задачи в получении максимальной производительности и покорении максимальной планки FPS. Рекомендуется установить всем, кто не хочет сильно париться с компиляцией и настройкой других ядер. Ядро можно установить из репозиториев, ибо оно имеет официальную поддержку дистрибутивом.

Из основных улучшений:

- Улучшено поведение планировщика CFS для значительного снижения задержек в несколько раз.
- Задействован эффективный алгоритм сжатия LZ4 для файла подкачки через zswap.
- Используется планировщик ввода/ввода (IO) BFQ, имеющий более низкие задержки и большую пропускную способность (Подробнее - [тут](#)).
- Ядро приспособлено к сохранению качества отклика системы при высокой нагрузке.

#### **I. Установка**

```
sudo pacman -S linux-zen linux-zen-headers

# Если у вас не GRUB - используйте команду обновления вашего загрузчика
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

#### **II. Установка (нативная компиляция)**

```
git clone https://aur.archlinux.org/linux-zen-git.git
cd linux-zen-git
makepkg -sric
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

### **7.1.2 liquorix**

linux-lqx - является по сути тем же Zen, но заточенным под Debian системы. Несмотря на это, авторы ARU считают его лучшим ядром на Arch Linux, по крайне мере для процессоров Intel. При нативной компиляции идеально подходит для игр в связке с wine-tkg.

Содержит множество изменений не вошедших в Zen (в виду разных причин):

- Планировщик PDS по умолчанию (На выбор предоставляются также BMQ и CFS), имеющий лучшие результаты в повседневных задачах.
- Алгоритм TCP BBR2 обеспечивающий более высокую скорость работы и максимизирует пропускную способность
- Multigenerational LRU для сохранения качества отклика в условиях нехватки оперативной памяти.
- 1000Hz тактовая частота по умолчанию для более точного планирования с минимумом зависаний.

- Предпочтительно жёсткое вытеснение процессов из очереди. Это гарантирует вам то, что не один процесс не сможет выжрать все процессорное время (дать системе зависнуть).
- Содержит минимальное количество отладочных функций.
- Другие внутренние изменения ядра.

При этом наследует все изменения из Zen.

## I. Установка (бинарные пакеты)

```
sudo pacman-key --keyserver https://keyserver.ubuntu.com --recv-keys 9AE4078033F8024D
sudo pacman-key --lsign-key 9AE4078033F8024D      # Добавляем GPG ключ
sudo nano /etc/pacman.conf

# Прописываем эти две строчки в файл
[liquorix]
Server = https://liquorix.net/archlinux/$repo/$arch
```

```
almarus@almarus:~ [1/1] /etc/pacman.conf Изменён
#Include = /etc/pacman.d/mirrorlist

[community]
Include = /etc/pacman.d/mirrorlist

# If you want to run 32 bit applications on your x86_64 system,
# enable the multilib repositories as required here.

#[multilib-testing]
#Include = /etc/pacman.d/mirrorlist

[multilib]
Include = /etc/pacman.d/mirrorlist

[liquorix]
Server = https://liquorix.net/archlinux/$repo/$arch

# An example of a custom package repository. See the pacman manpage for
# tips on creating your own repositories.
#[custom]

^K Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^M Замена ^U Вставить ^J Выровнять ^/ К строке
```

```
sudo pacman -Suuuyy
sudo pacman -S linux-lqx linux-lqx-headers
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

Вариант установки I рекомендуется если не хотите компилировать, но тогда производительность будет хуже чем у аналогичного скомпилированного ядра.

## II. Установка и настройка

В этом случае мы настроим ядро и выполним его нативную-компиляцию. Тонкая настройка ядра позволит дать ещё больше производительности и может ускорить сам процесс сборки.

```
git clone https://aur.archlinux.org/linux-lqx.git          # Скачивание
↳ исходников.
cd linux-lqx                                         # Переход в linux-
↳ lqx
gpg --keyserver keyserver.ubuntu.com --recv-keys 38DBBDC86092693E # GPG ключ
sed -i 's/_makenconfig=/_makenconfig=y/' PKGBUILD           # Включаем ручную
↳ настройку
makepkg -srcc
```

После некоторого времени с началом компиляции перед вами предстанет окно с настройкой ядра. Подробные инструкции и рекомендации по настройке вы найдете в следующем разделе.



### 7.1.3 Xanmod

Альтернатива liquorix, так же нацеленная на оптимизацию под игрушки и повышение плавности работы системы. Новомодное ядро, которое включает в себе часть уже описанных выше изменений из zen/lqx. Помимо прочего имеет:

- Улучшения производительность протокола TCP (включая BBRv2)
- Частично включает в себя патчи от Clear Linux (также как и linux-lqx)
- WineSync, альтернатива Fsync, ещё одна реализация синхронизации NT примитов для Wine, но вынесенная в качестве отдельного драйвера (модуля). Для остальных ядер может быть установлена через AUR пакет [winesync-dkms](#).
- Высокая пропускная способность устройств ввода/вывода.
- Улучшения систем кэширования и управления памятью.

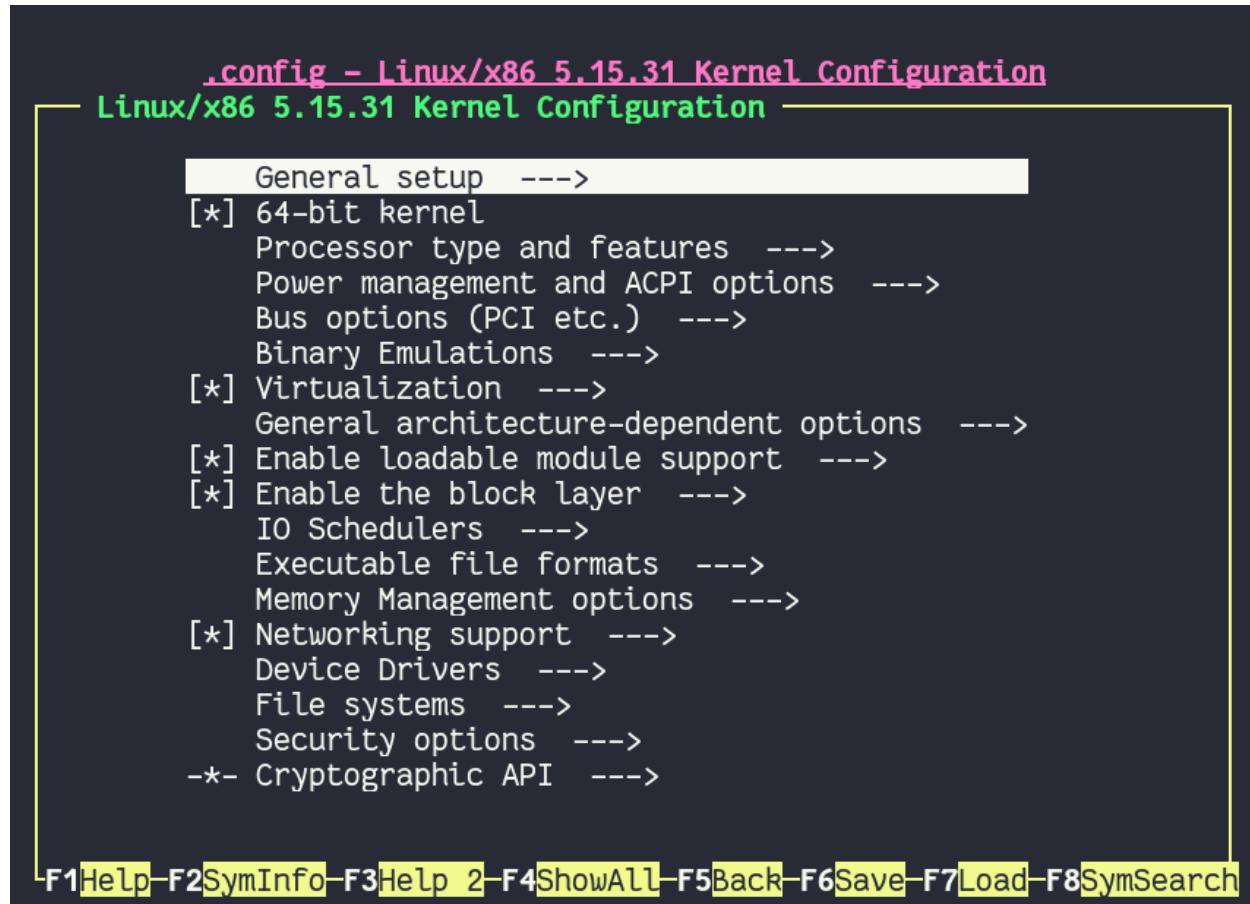
Полный список включаемых в него патчей вы можете найти здесь: <https://github.com/xanmod/linux-patches>

**Внимание:** Не рекомендуется обладателям процессоров Intel, т.к. возможно все ещё имеет нерешенную проблему со сбросом частот процессора от данного производителя (<https://forum.xanmod.org/thread-3800.html>)

#### I. Установка (компиляция):

```
git clone https://aur.archlinux.org/linux-xanmod.git           # Скачивание
↳ исходников.
cd linux-xanmod                                              # Переход в
↳ linux-xanmod
gpg --keyserver hkp://keyserver.ubuntu.com --recv-keys 38DBBDC86092693E # GPG ключ
export _makenconfig=y _use_numa=n _use_tracers=n _compiler=clang      # Включаем
↳ ручную настройку
makepkg -srcc                                             # Сборка и
↳ установка
```

После некоторого времени с начала сборки у вас должно появится окно с ручной настройкой ядра. Этот процесс мы подробнее рассмотрим в следующей главе.



### 7.1.4 linux-tkg

Является альтернативой всем трем ядрам выше, что предоставляет возможность собрать ядро с набором множества патчей на улучшение производительности в игрушках (Futex2, Zenify). Предоставляет выбор в сборке ядра с разными планировщиками. Грубо говоря, это ядро сборная солянка из всех остальных ядер с большим набором патчей.

#### I. Установка и настройка:

```
git clone https://github.com/Frogging-Family/linux-tkg.git  
cd linux-tkg
```

Есть две возможности предварительной настройки `linux-tkg`: либо через редактирование файла `customization.cfg`, либо через терминал по ходу процесса установки. Мы выбираем первое и отредактируем `customization.cfg`:

```
nano customization.cfg
```

Итак, настройка здесь достаточно обширная поэтому мы будем останавливаться только на интересующих нас настройках:

`_version="5.17"` - Здесь выбираем версию ядра которую мы хотим установить. Выбирайте самую последнюю из доступных.

`_modprobeddb="false"` - Опция отвечающая за сборку мини-ядра. Подробнее о нем вы

можете узнать в соответствующем разделе. Если хотите собрать мини-ядро - пишите "true", если нет - "false".

`_menuconfig="2"` - Выбор настройки ядра через menuconfig/xconfig/nconfig. Рекомендуется выбрать "2" чтобы перед сборкой можно было выполнить непосредственную настройку ядра через menunconfig как мы уже делали ранее с liquorix.

`_cpusched="pds"` - Выбор CPU планировщика ядра. К выбору предоставляется довольно много планировщиков, но мы советуем обратить ваше внимание только на некоторых из них: "pds", "bmq", "cacule", "cfs" (дефолтный для ванильного ядра). По некоторым данным, PDS дает больше FPS, а CacULE дает лучшие задержки по времени кадра (плавность). Однако все слишком ситуативно чтобы выбрать из них лучшего, в каких-то играх/задачах будет выигрывать PDS, а в каких-то CacULE и так далее.

Рекомендуется попробовать PDS или CacULE.

`_rr_interval="default"` - Задает продолжительность удержания двумя задачами одинакового приоритета. Рекомендуемое значение слишком зависит от выбранного планировщика, поэтому лучше всего задавайте "default".

`_default_cpu_gov="performance"` - Выбирает режим по умолчанию в котором будет масштабироваться частота процессора. Рекомендуется "performance" чтобы процессор по умолчанию работал в режиме высокой производительности.

`_aggressive_ondemand="false"` - Задает агрессивное применение динамического управления частотой процессора по необходимости в выполняемой задаче, обеспечивая тем самым энергоэффективность. Но т.к. выше мы уже закрепили режим масштабирования "performance", то мы можем отключить этот параметр. Однако пользователи ноутбуков могут оставить этот параметр включенным.

`_disable_acpi_cpufreq="true"` - Отключает универсальный acpi\_freq драйвер масштабирования частоты процессора в угоду фирменному драйверу Intel/AMD процессоров что имеют лучшую производительность по сравнению с acpi\_freq. Выбирайте значение по собственному усмотрению со знанием своего CPU.

`_sched_yield_type="0"` - Настраивает выполнение освобождения процесса от потребления процессорного времени путем его переноса в конец очереди выполнения процессов. Рекомендуемое значение для лучшей производительности - "0", т.е. не осуществлять перенос в конец очереди для освобождения процесса.

`_tickless="0"` - Рекомендуется выбирать статические тики таймера ядра.

`_timer_freq="1000"` - Задает частоту таймера. Рекомендуется 1000 для лучшей отзывчивости системы на домашнем ПК или ноутбуке.

`_fsync="true"` - Задействует поддержку ядром замены Esync от компании Valve - Fsync. Обязательно к включению ("true") для лучшей производительности в играх.

`_futex2="true"` - Осуществляет использование нового, экспериментального futex2 вызова что может дать лучшую производительность для игрушек запускаемых через Wine/Proton. Для обычных ядер поддержка Futex2 включена начиная с версии 5.16+.

`_winesync="false"` - Ещё одна замена esync, но уже от разработчиков Wine.

`_zenify="true"` - Применяет твики Zen и Liquorix для улучшения производительности в играх. Настоятельно рекомендуется к включению.

`_complierlevel="1"` - Задает степень оптимизации ядра во время сборки. Лучше всего выбирать "1", т.е. сборку с -O2 флагом (высокая производительность).

`_processor_opt="native_intel"` - С учетом какой архитектуры процессора собирать ядро. Настоятельно рекомендуется указать здесь либо архитектуру непосредственно ва-

шего процессора (К примеру: "skylake"), либо фирму производитель, где для Intel это - "native\_intel", для AMD - "native\_amd".

`_ftrcdisable="true"` - Отключает лишние трекеры для отладки ядра.

`_acs_override="true"` - Включает патч на разделение сгруппированных PCI устройств в IOMMU, которые могут понадобиться вам отдельно. По умолчанию есть в linux-zen и linux-lqx. Подробнее читайте - [здесь](#). Советуем включить если в будущем вы хотите выполнить операцию прогона вашей видеокарты в виртуальную машину.

Вот и все. Остальные настройки *customization.cfg* вы можете выбрать по собственному предпочтению. После того как мы закончили с настройкой, можно перейти непосредственно к сборке и установке ядра::

```
makepkg -srcc # Сборка и установка linux-tkg
```

## 7.2 Настройка ядра

При нативной компиляции ядра обязательным этапом является его настройка. Хотя и заботливые сопровождающие кастомных ядер обычно уже заранее выполняют всю работу за вас, есть пара моментов на которых стоит остановиться.

После начала компиляции через некоторое время перед вами появится меню настройки ядра. Перемещение между пунктами в нем осуществляется стрелками на клавиатуре, переход в раздел через клавишу *Enter*, а выход из него через *Esc*.

Здесь нужно следовать графической инструкции.

**1.** Для начала выберем одну из важнейших настроек. Это выбор архитектуры процессора под которую будет компилироваться ядро. По умолчанию выбрана *Generic*, то есть такое ядро будет собираться под абстрактную x86 архитектуру и при этом будет совместимо с любым процессором, хоть AMD, хоть Intel. Главным же преимуществом именно нативной компиляции любого ПО состоит в задействовании максимума производительности конкретно под вашу архитектуру процессора, с использованием всех доступных ему инструкций. А в случае с ядром это особенно важно, ибо ядро это сердце операционной системы, и если его нативно собрать под себя, то мы получаем существенный прирост и отличный отклик системы. Поэтому начиная с главного окна настройки перейдите в раздел "*Processor type and features*" и затем в "*Processor family*". Здесь выберите либо "*Intel-native optimizations*" если у вас процессор Intel, либо "*AMD-native optimizations*" если у вас процессор AMD, как это показано на скриншотах ниже.

### 1.1

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Linux/x86 5.16.17-lqx1 Kernel Configuration

General setup --->
[*] 64-bit kernel
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Binary Emulations --->
[*] Virtualization --->
    General architecture-dependent options --->
[*] Enable loadable module support --->
-`- Enable the block layer --->
    Executable file formats --->
    Memory Management options --->
[*] Networking support --->
    Device Drivers --->
    File systems --->
    Security options --->
-`- Cryptographic API --->
    Library routines --->

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

1.2

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Processor type and features

[*] Symmetric multi-processing support
-*= Support x2apic
[*] Enable MPS table
[*] Avoid speculative indirect branches in kernel
[*] x86 CPU resource control support
[ ] Support for extended (non-PC) x86 platforms
[*] Intel Low Power Subsystem Support
[*] AMD ACPI2Platform devices support
-*= Intel SoC IOSF Sideband support for SoC platforms
[ ]   Enable IOSF sideband access through debugfs
[*] Single-depth WCHAN output
[*] Linux guest support --->
      Processor family (Intel-Native optimizations autodetected)
[ ] Old AMD GART IOMMU support
      (128) Maximum number of CPUs
[ ] Cluster scheduler support
[*] Multi-core scheduler support
[*]   CPU core priorities scheduler support

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

1.3

```

.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Processor family

Intel Haswell
Intel Broadwell
Intel Skylake
Intel Skylake X
Intel Cannon Lake
Intel Ice Lake
Intel Cascade Lake
Intel Cooper Lake
Intel Tiger Lake
Intel Sapphire Rapids
Intel Rocket Lake
Intel Alder Lake
Generic-x86-64
Generic-x86-64-v2
Generic-x86-64-v3
Generic-x86-64-v4
<X> Intel-Native optimizations autodetected by the compiler
      AMD-Native optimizations autodetected by the compiler

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch

```

**(Важно:** автор выбрал здесь Intel-native, но **если у вас процессор от AMD выбирайте только AMD-native )**

2. Изменим поведение таймера ядра. Дело в том, что ядро может осуществлять прерывания для перепланирования задач процессора либо статически через частоту N (один тик), либо динамически. Динамический таймер работает только тогда, когда процессор находится в работе, т. е. когда процессор простоявает таймер прерываний останавливает свою работу (из-за ненадобности). Существует также и вариант динамического таймера когда тики не происходят даже тогда, когда процессор чем-то занят.

Собственно выбор этих трех вариантов и дан нам на скриншотах ниже, где:

- Periodic timer ticks - осуществление тика статически через частоту N
- Idle dynticks system - прерывания через частоту тика N только тогда, когда процессор чем-то занят.
- Full dynticks system - прерывания через частоту тика N, но не всегда, даже если процессор чем-то занят.

### Что из этого выбрать?

По нашему мнению динамические тики лучше всего выбирать тем людям, которые хотят уменьшить энергопотребление процессора. Т.е. всем пользователям ноутбуков/нетбуков посвящается. Обратите внимание, что *Full dynticks system* может несколько ухудшить производительность в зависимости от железа, но даёт реальные преимущества в экономии энергии.

Но если у вас домашний стационарный ПК или вам просто все равно на энергопотреб-

ление - лучше выбрать статические тики ("Pereodic timer ticks"), т.к. они потенциально дают более предсказуемый метод планирования прерываний. Это значит, что теоретически у вас не будет тратиться время на выход таймера из "сна". И на практике оказывается что периодические тики дают лучшую производительность в играх и мультимедиа.

## 2.1

.config - Linux/x86 5.16.17-lqx1 Kernel Configuration  
Linux/x86 5.16.17-lqx1 Kernel Configuration

```
General setup --->
[*] 64-bit kernel
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Binary Emulations --->
[*] Virtualization --->
    General architecture-dependent options --->
[*] Enable loadable module support --->
-`- Enable the block layer --->
    Executable file formats --->
    Memory Management options --->
[*] Networking support --->
    Device Drivers --->
    File systems --->
    Security options --->
-`- Cryptographic API --->
    Library routines --->

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

## 2.2

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
General setup

[*] Tune kernel for interactivity
[ ] Compile also drivers which will not load
[ ] Compile the kernel with warnings as errors
    () Local version - append to kernel release
[ ] Automatically append version information to the version string
    () Build ID Salt
    Kernel compression mode (ZSTD) --->
    () Default init path
    ((none)) Default hostname
[*] Support for paging of anonymous memory (swap)
[*] System V IPC
[*] POSIX Message Queues
[*] General notification queue
[*] Enable process_vm_readv/writev syscalls
[ ] uselib syscall
--*- Auditing support
    IRQ subsystem --->
        Timers subsystem --->

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

2.3

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Timers subsystem

    Timer tick handling (Periodic timer ticks (constant rate,
[ ] Old Idle dynticks config
[*] High Resolution Timer Support

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

2.4

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Timer tick handling

<X> Periodic timer ticks (constant rate, no dynticks)
    Idle dynticks system (tickless idle)
    Full dynticks system (tickless)

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

3. Просим вас удостовериться в значениях частоты таймера. Это как раз то самое N через которое происходит тик таймера и последующее за ним прерывание. Рекомендуемое значение для домашнего ПК/Ноутбука это 1000. Однако если вы имеете многоядерный процессор (12 и более потоков) или какой-нибудь серверный Intel Xeon, то вы можете попробовать установить частоту ниже 1000.

### 3.1

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Linux/x86 5.16.17-lqx1 Kernel Configuration

    General setup --->
[*] 64-bit kernel
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Binary Emulations --->
[*] Virtualization --->
    General architecture-dependent options --->
[*] Enable loadable module support --->
-*= Enable the block layer --->
    Executable file formats --->
    Memory Management options --->
[*] Networking support --->
    Device Drivers --->
    File systems --->
    Security options --->
-*= Cryptographic API --->
    Library routines --->

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

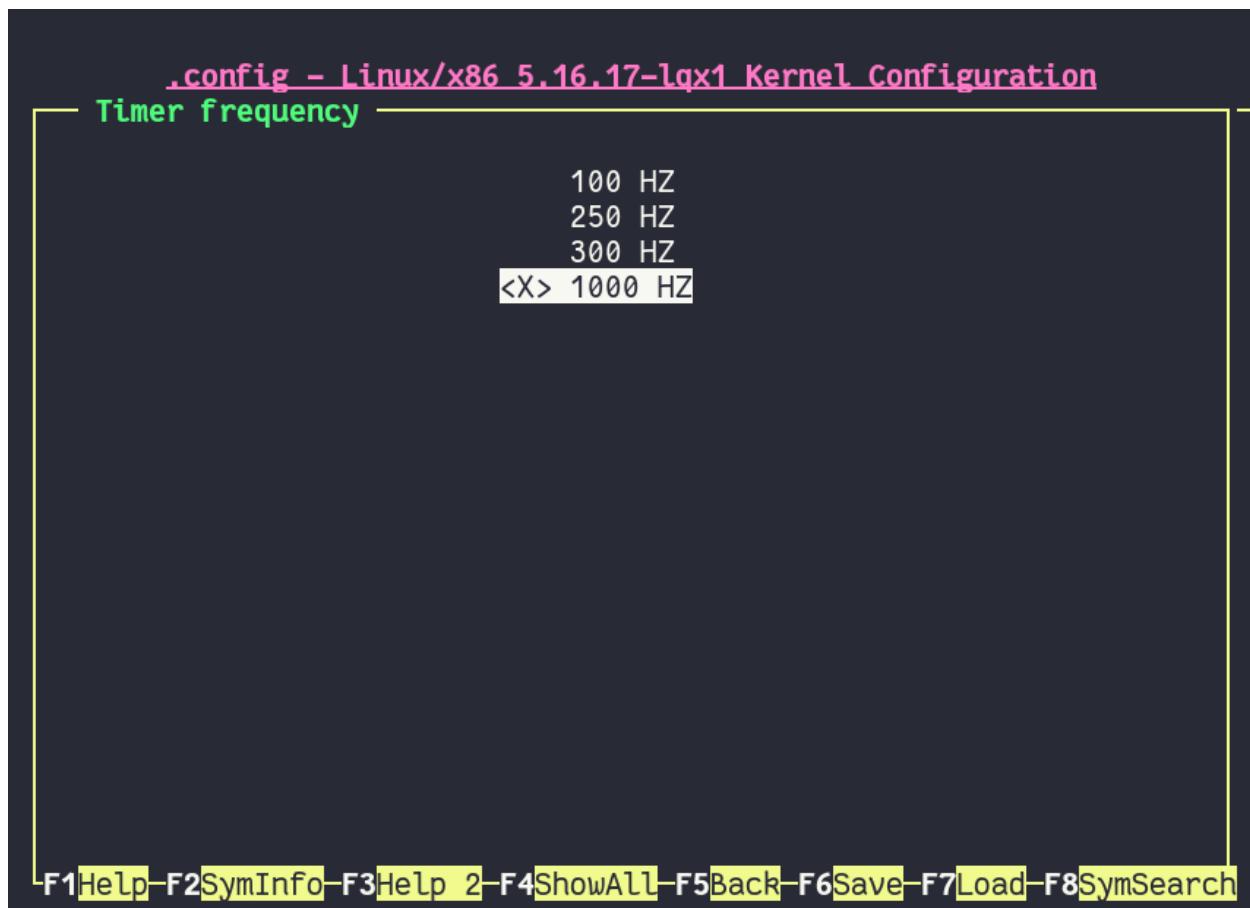
3.2

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Processor type and features

-- MTRR (Memory Type Range Register) support
[*] MTRR cleanup support
    (1) MTRR cleanup enable value (0-1)
    (0) MTRR cleanup spare reg num (0-7)
[*] Memory Protection Keys
    TSX enable mode (auto) --->
[*] Software Guard eXtensions (SGX)
[*] EFI runtime service support
[*]   EFI stub support
[*]   EFI mixed-mode support
    Timer frequency (1000 HZ) --->
[*] kexec system call
[ ] kexec file based system call
[ ] kernel crash dumps
[*] kexec jump
-- Build a relocatable kernel
[ ] Randomize the address of the kernel image (KASLR)
    (0x1000000) Alignment value to which kernel should be alig

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

3.3



4. Рекомендуем вам отключать отладочные функции ядра. Они тоже имеют свои накладные расходы и в большинстве случаев вы ими пользоваться никогда не будете, а на крайний случай у вас должно быть установлено ядро linux-lts как запасной аэродром. Для их отключения из главного меню перейдите в "Kernel Hacking" и сделайте там все так, как показано на скриншотах:

---

**Примечание:** Обращаем ваше внимание на то, что на некоторых ядрах не все возможные отладочные параметры могут быть отключены. Например Xanmod не позволяет отключить ряд параметров из списка ниже. Но вы можете ими пренебречь.

---

#### 4.1

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Linux/x86 5.16.17-lqx1 Kernel Configuration

[*] 64-bit kernel
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Binary Emulations --->
[*] Virtualization --->
    General architecture-dependent options --->
[*] Enable loadable module support --->
-*= Enable the block layer --->
    Executable file formats --->
    Memory Management options --->
[*] Networking support --->
    Device Drivers --->
    File systems --->
    Security options --->
-*= Cryptographic API --->
    Library routines --->
        Kernel hacking --->

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

4.2

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Kernel hacking

    printk and dmesg options --->
    Compile-time checks and compiler options --->
    Generic Kernel Debugging Instruments --->
    [ ] Kernel debugging
        Memory Debugging --->
        Debug Oops, Lockups and Hangs --->
        Scheduler Debugging ----
    [ ] Enable extra timekeeping sanity checking
        Lock Debugging (spinlocks, mutexes, etc...) --->
    [ ] Debug IRQ flag manipulation
    [ ] Stack backtrace support
    [ ] Warn for all uses of unseeded randomness
        Debug kernel data structures --->
        RCU Debugging --->
    [ ] Tracers ----
    [ ] Remote debugging over FireWire early on boot
    [ ] Sample kernel code ----
    [*] Filter access to /dev/mem

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

5. Обладателям видеокарт NVIDIA советуем отключить поддержку фирменного фреймбуфера, как бы странно это не звучало. Это позволит вам избежать проблемы конфликта фреймбуфера ядра и фреймбуфера бинарного драйвера NVIDIA. Сделайте это как показано ниже:

## 5.1

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Linux/x86 5.16.17-lqx1 Kernel Configuration

    General setup --->
[*] 64-bit kernel
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Binary Emulations --->
[*] Virtualization --->
    General architecture-dependent options --->
[*] Enable loadable module support --->
-`*- Enable the block layer --->
    Executable file formats --->
    Memory Management options --->
[*] Networking support --->
    Device Drivers --->
        File systems --->
        Security options --->
-`*- Cryptographic API --->
        Library routines --->

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

5.2

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Device Drivers

[*] Board level reset or power off  --->
-*= Power supply class support  --->
{*} Hardware Monitoring support  --->
-*= Thermal drivers  --->
[*] Watchdog Timer Support  --->
{M} Sonics Silicon Backplane support  --->
{M} Broadcom specific AMBA  --->
    Multifunction device drivers  --->
-*= Voltage and Current Regulator Support  --->
<M> Remote Controller support  --->
    CEC support  --->
<M> Multimedia support  --->
    Graphics support  --->
<M> Sound card support  --->
    HID support  --->
[*] USB support  --->
<M> MMC/SD/SDIO card support  --->
<M> Sony MemoryStick card support  --->

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

5.3

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Graphics support

< > Simple framebuffer driver
<M> DRM support for HX8357D display panels
<M> DRM support for ILI9225 display panels
<M> DRM support for ILI9341 display panels
<M> DRM support for ILI9486 display panels
<M> DRM support for MI0283QT
<M> DRM support for Pervasive Displays RePaper panels (V231)
<M> DRM support for Sitronix ST7586 display panels
<M> DRM support for Sitronix ST7715R/ST7735R display panels
<M> Para-virtualized frontend driver for Xen guest OS
<M> Virtual Box Graphics Card
<M> GUD USB Display
<M> DRM Support for Hyper-V synthetic video device
[ ] Enable legacy drivers (DANGEROUS) ----
    Frame buffer Devices --->
        Backlight & LCD device support --->
        Console display driver support --->
[ ] Bootup logo ----

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

5.4

```
.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Frame buffer Devices ━━━━━━
    <*> Support for frame buffer devices  --->
F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch
```

5.5

```

.config - Linux/x86 5.16.17-lqx1 Kernel Configuration
Support for frame buffer devices

[*] VESA VGA graphics support
[*] EFI-based Framebuffer Support
<M> N411 Apollo/Hecuba devkit support
<M> Hercules mono graphics support
< > OpenCores VGA/LCD core 2.0 framebuffer support
< > Epson S1D13XXX framebuffer support
< > nVidia Framebuffer Support
< > nVidia Riva support
< > Intel740 support
<M> Intel LE80578 (Vermilion) support
<M> Intel Carillo Ranch support
<M> Matrox acceleration
[*] Millennium I/II support
[*] Mystique support
[*] G100/G200/G400/G450/G550 support
<M> Matrox I2C support
<M> G400 second head support
<M> ATI Radeon display support

F1Help-F2SymInfo-F3Help 2-F4ShowAll-F5Back-F6Save-F7Load-F8SymSearch

```

## 7.3 Сборка ядра с помощью Clang + LTO

В разделе "Общее ускорение системы" мы уже говорили о преимуществах сборки пакетов при помощи компилятора Clang вместе с LTO оптимизациями. Но ядро требует отдельного рассмотрения, ибо те параметры которые мы указали ранее в makepkg.conf не работают для сборки ядра, и по прежнему будут применяться компиляторы GCC.

Чтобы активировать сборку ядра через Clang нужно:

- Для ядра linux-xanmod экспортировать данную переменную окружения перед выполнением команды сборки: `export _compiler=clang`
- Для ядра linux-tkg в конфигурационном файле `customization.cfg` включить параметр `_compiler="llvm"` (В том же файле можно настроить применение LTO оптимизаций через параметр `_lto_mode`. О режимах LTO читайте далее).
- Для всех остальных ядер, устанавливаемых из AUR (включая linux-lqx), нужно просто экспортировать переменные окружения `LLVM=1` и `LLVM_IAS=1` перед командой сборки:

```

export LLVM=1 LLVM_IAS=1 # Без переменной LLVM_IAS станет невозможным применение
                        # LTO оптимизаций
makepkg -srcc           # Сборка и установка желаемого ядра

```

Теперь перейдем к настройке LTO оптимизаций. Для этого на этапе конфигурации вального ядра зайдите в "General architecture-dependent options" -> "Link Time Optimization (LTO)" как показано на изображениях:

1.

```
.config - Linux/x86 5.15.12-lqx1 Kernel Configuration

Linux/x86 5.15.12-lqx1 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

General setup --->
[*] 64-bit kernel
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Binary Emulations --->
[*] Virtualization --->
[*] General architecture-dependent options --->
[*] Enable loadable module support --->
-** Enable the block layer --->
  IO Schedulers --->
  Executable file formats --->
  Memory Management options --->
[*] Networking support --->
  Device Drivers --->
  File systems --->
v(+)

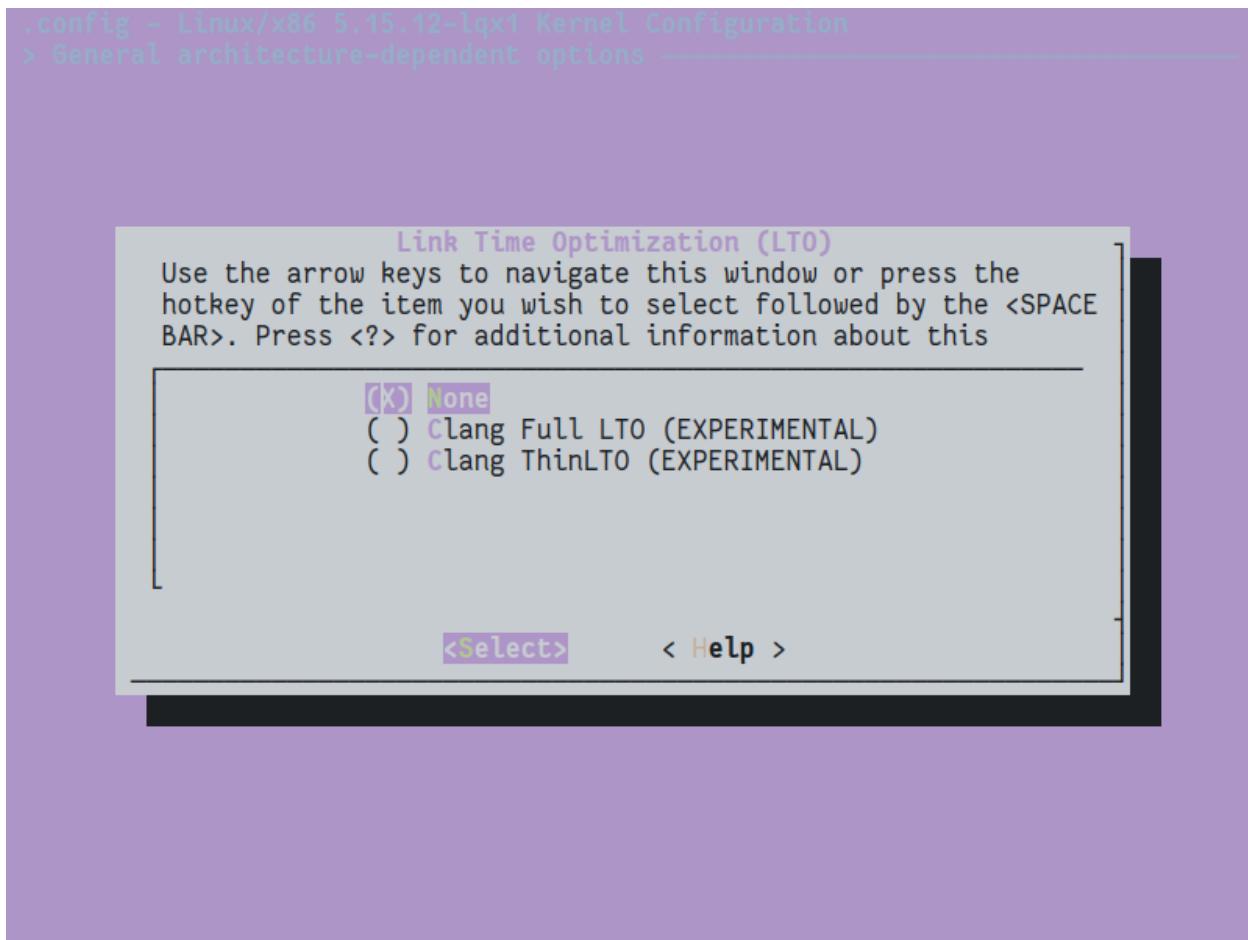
<Select>  < Exit >  < Help >  < Save >  < Load >
```

2.

```
.config - Linux/x86 5.15.12-lqx1 Kernel Configuration
> General architecture-dependent options
    General architecture-dependent options
        Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
        submenus ----). Highlighted letters are hotkeys. Pressing <Y>
        includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
        exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
            [*] Kprobes
            [*] Optimize very unlikely/likely branches
            [ ] Static key selftest
            [ ] Static call selftest
            [*] Enable seccomp to safely execute untrusted bytecode
            [ ] Show seccomp filter cache status in /proc/pid/seccomp_cache
            [*] Stack Protector buffer overflow detection
            [*] Strong Stack Protector
            Link Time Optimization (LTO) (None) --->
            [*] Provide system calls for 32-bit time_t
            [*] Use a virtually-mapped stack
            [ ] Randomize kernel stack offset on syscall entry
            [ ] Locking event counts collection
            GCOV-based kernel profiling --->

        <Select>      < Exit >     < Help >     < Save >     < Load >
```

3.



На последнем изображении показано окно выбора режима применения LTO оптимизации. Этих режимов всего два:

1. Полный (Full): использует один поток для линковки, во время сборки медленный и использует больше памяти, но теоретически имеет наибольший прирост производительности в работе уже готового ядра.
2. Тонкий (Thin): работает в несколько потоков, во время сборки быстрее и использует меньше памяти, но может иметь более низкую производительность в итоге чем Полный (Full) режим.

Мы рекомендуем использовать "Полный (Full)" режим чтобы получить в итоге лучшую производительность.

**Внимание:** Сборка ядра через Clang работает только с версией ядра 5.12 и выше!

Больше подробностей по теме вы можете найти в данной статье:

<https://habr.com/ru/company/ruvds/blog/561286/>

# ГЛАВА 8

---

## Wine / Linux Gaming

---

### 8.1 Основные составляющие

Переходя к запуску Windows-игр на Linux-системах, стоит иметь в виду, что никаких эмуляторов Windows на Linux не существует, и весь запуск осуществляется с помощью сторонней реализации Windows API — Wine/Proton, а также средств ретрансляции команд DirectX в доступные графические API на Linux (Vulkan, OpenGL) с помощью DXVK или иных ретранслятора кода.

#### 8.1.1 Что такое Wine?

Wine - слой совместимости для запуска Windows-приложений (в том числе игр) из под Linux (Unix-подобных систем). Благодаря нему вы по факту сможете поиграть в большинство игр из вашей библиотеки Steam/GOG/Epic Games Store. Исключением разве что являются игры с встроенными анти-чит системами, хотя благодаря усилиям Valve, в ближайшем будущем, вероятно, это уже не будет являться такой большой проблемой. Конечно, все не так гладко как хотелось бы, ведь для запуска и обеспечения работоспособности многих программ/игр придется ещё изрядно повозиться с его настройкой, однако сама такая возможность в принципе является незаменимой для Linux пользователей, в частности геймеров.

## 8.1.2 Сборки Wine

Существуют различные сборки Wine. Подобный зоопарк появился ввиду накопления большого количества различных патчей (сторонних изменений) которые по какой-то причине не могут быть приняты в основную ветку разработки Wine. Кроме того, стоит понимать что, как и в случае с ядрами, обычный Wine это прежде всего свободная реализация Windows API, которая подразумевает запуск любых Windows приложений. При этом он не заточен конкретно под игры или любой другой софт. Именно поэтому в том числе и появились такие вещи как Proton от компании Valve, являющийся по сути тем же Wine, но с упором именно на игровую составляющую, исправляющий многие проблемы обычного Wine связанные с играми.

На текущий момент есть две официальных сборки Wine которые поддерживаются непосредственно разработчиками:

- wine - обычная, стабильная версия, содержащая только проверенные изменения от разработчиков, и которая условно универсальна для любых приложений.
- wine-staging - содержащая те изменения которые пока не могут попасть в обычную версию, но которые могут помочь исправить определенные баги и улучшить работу конкретных программ и частей Wine.

Но существуют также много альтернативных сборок основанных на Wine-staging с упором именно на игры, о них написано далее.

### 8.1.2.1 Установка wine-staging вместе с зависимостями

Бинарные версии ПО всегда доступны в репозиториях и очень удобны, но они не могут обеспечить достойный уровень производительности. Для начала советую поставить wine-staging вместе со всеми зависимостями, а уже затем собрать wine-tkg.

```
sudo pacman -S wine-staging winetricks wine-mono giflib lib32-giflib libpng lib32-
→ libpng libldap lib32-libldap gnutls lib32-gnutls mpg123 lib32-mpg123 openal lib32-
→ openal v4l-utils lib32-v4l-utils libpulse lib32-libpulse libgpg-error lib32-libgpg-
→ error alsal-plugins lib32-alsa-plugins alsalib lib32-alsa-lib libjpeg-turbo lib32-
→ libjpeg-turbo sqlite lib32-sqlite libxcomposite lib32-libxcomposite libxinerama
→ lib32-libgcrypt libgcrypt lib32-libxinerama ncurses lib32-ncurses opencl-icd-loader
→ lib32-opencl-icd-loader libxslt lib32-libxslt libva lib32-libva gtk3 lib32-gtk3 gst-
→ plugins-base-libs lib32-gst-plugins-base-libs vulkan-icd-loader lib32-vulkan-icd-
→ loader
```

## 8.1.3 Альтернативные сборки Wine

По умолчанию обычные сборки Wine недостаточно хорошо заточены для комфортной игры ввиду их универсальности, т.к. это все таки свободная реализация WinAPI в Linux и она не обязана использоваться только для запуска игр из под Windows в Linux. Но существуют также альтернативные сборки Wine, с большим количеством различных патчей и улучшений, нацеленных в основном как раз на игры.

### 8.1.3.1 WINE-TKG

WINE-TKG - это, наверное, лучшая сборка Wine для опытных пользователей которые хотят улучшить свой опыт игры под линуксом. Преимуществом данной сборки перед другими является огромное количество вложенных в неё патчей из разных источников (В том числе портированных из Proton). Поэтому мы настоятельно рекомендуем её к установке если вы хотите получить не только больше производительности, но и совместимости с различными Windows играми.

Установку wine-tkg можно выполнить двумя способами:

- I. Установить из его PKGBUILD как мы это делал ранее с другими программами.
- II. Собрать его полностью вручную из исходников.

Мы выберем первый вариант установки, т.к. он самый простой и надежный.

Второй вариант вы можете осуществить по желанию, особенно если у вас дистрибутив отличный от Arch Linux.

#### I. Установка

```
git clone https://github.com/Frogging-Family/wine-tkg-git.git
cd wine-tkg-git/wine-tkg-git
```

По аналогии с linux-tkg, wine-tkg предоставляет возможность предварительно настроить себя перед установкой на применение различных патчей и твиков через редактирование файла *customization.cfg*:

```
nano customization.cfg
```

Здесь нас интересует не так много настроек. По сути можете оставлять все значения по умолчанию, кроме следующих параметров:

`_use_esync="true"` - Включает поддержку esync что оптимизирует работу wineserver. Активируется через переменную окружения `WINEESYNC=1`.

`_use_fsync="true"` - Включает поддержку fsync, альтернативу esync которую можно задействовать через переменную окружения `WINEFSYNC=1`. Оба параметра обязательны к включению для повышения производительности.

`_fsync_futex2="true"` - Включает поддержку futex2 для fsync. Улучшает производительность игр при использовании Fsync. Требуется ядро версии 5.16 и выше.

Подробное сравнение Esync и Fsync можно посмотреть в данном видео.

[https://www.youtube.com/watch?v=-nlNJguG5\\_0&t=18s](https://www.youtube.com/watch?v=-nlNJguG5_0&t=18s)

`_launch_with_dedicated_gpu="false"` - Активирует запуск приложений через дискретный графический процессор на ноутбуках с PRIME. Работает только с открытыми драйверами (Mesa), поэтому выбирайте сами нужно оно вам или нет.

`_update_winevulkan="true"` - Включает свежие обновления библиотеки winevulkan. Обязательно оставляйте включенным.

`_FS_bypass_compositor="true"` - Задействует обход композитора приложениями запускаемыми через Wine. Очень полезная и нужная опция для исправления проблем задержек и заиканий в играх, в случае когда системный композитор пытается лишний раз осуществить композитинг над окном с игрой запущенной через Wine. Обязательно включаем.

`_proton_fs_hack="true"` - Включает еще один очень нужный патч. Вносит исправление с помощью которого изменяя разрешение игры в полноэкранном режиме у вас не будет изменяться разрешение вашего рабочего стола. Включаем.

`_msvcrt_nativebuiltin="true"` - Осуществляет нативную сборку msrv.dll. Лишним точно не будет, поэтому включаем.

`_win10_default="false"` - Устанавливает в качестве версии по умолчанию Windows 10 в Wine. Не рекомендуется к включению ввиду того, что это может задействовать vkd3d в некоторых играх работающих на DirectX 12, что однако ведет к ухудшению производительности по сравнению с DXVK при возможности запустить игру с DirectX 11.

`_protonify="true"` - Задействует множественные заплатки и патчи для Wine портированные из Proton. По нашему мнению это масть хев, т.к. они содержат в себе множественные исправления для различных игр и оптимизаций к ним. Настоятельно рекомендуется к включению.

**Внимание:** По умолчанию wine-tkg не использует нативные флаги которые вы указывали ранее в `/etc/makepkg.conf`. Их нужно указать вручную отредактировав `wine-tkg-profiles/advanced-customization.cfg`:

```
nano wine-tkg-profiles/advanced-customization.cfg # Отредактируйте строчки ниже  
GCC_FLAGS="-O2 -ftree-vectorize -march=native"  
CROSS_FLAGS="-O2 -ftree-vectorize -march=native"
```

На этом все, остальные настройки оставьте по умолчанию.

Теперь можно перейти к самой сборке и установке wine-tkg: `makepkg -srcc`

## II. Ручная установка

Подробно описывать ручную сборку здесь мы не будем. Поэтому лучше всего посмотрите видео версию, где это наглядно показано (7 минута 23 секунда):

[https://www.youtube.com/watch?v=W1e6\\_3dPlHk](https://www.youtube.com/watch?v=W1e6_3dPlHk)

### 8.1.3.2 wine-tkg-userpatches

Это дополнение к wine-tkg. По сути это коллекция пользовательских патчей для улучшения производительности и совместности. Среди них: улучшения работы с памятью, интерфейсом GDI, качества отклика клавиатуры через системные вызовы Futex, повышение приоритета процессов Wine по умолчанию, и другие низкоуровневые изменения от сторонних разработчиков.

#### Установка:

```
git clone https://github.com/openglfreak/wine-tkg-userpatches  
cd ~wine-tkg-git/wine-tkg-git  
  
nano wine-tkg-profiles/advanced-customization.cfg # Отредактируйте строчку ниже  
EXT_CONFIG_PATH="~/wine-tkg-userpatches/wine-tkg.cfg"
```

Пересоберите wine-tkg по инструкции выше.

Никакой дополнительной настройки (редактирования *customization.cfg*) при этом не требуется.

### 8.1.3.3 Proton-GE-Custom

Proton-GE-Custom это форк проекта Proton для запуска Windows-игр с дополнительными патчами и оптимизациями не вошедшиими в основную ветку Proton, а также улучшение совместимости с некоторыми играми (например, Warframe). Позволяет играть во многие проекты которые не заводятся с обычным Wine или Proton.

#### I. Установка (бинарная версия)::

```
git clone https://aur.archlinux.org/proton-ge-custom-bin
cd proton-ge-custom-bin
makepkg -sric
```

#### II. Установка (компиляция, имеет много зависимостей)::

```
git clone https://aur.archlinux.org/proton-ge-custom
cd proton-ge-custom

# Нативные флаги + дополнительный патч для DXVK позволяющий улучшить производительность видеокарт NVIDIA
wget -qO proton-ge-aur.patch https://gist.githubusercontent.com/ventureoo/9b89c4799fbc89304f42983c6e90bda0/raw/9f10d463dfecaaa4935be757b48912004c6996fd/proton-ge-aur.patch PKGBUILD proton-ge-aur.patch

makepkg -sric
```

Дабы использовать Proton-GE в качестве альтернативы обычному Proton, после установки Proton-GE-Custom вам нужно перезапустить Steam и зайти в Свойства нужной вам игры, прожать в: *Совместность -> Принудительно использовать определенный инструмент совместности Steam Play -> Proton-6.XX-GE-1*. Готово, теперь можно запустить игру.

### 8.1.4 Использование Wine

Использование Wine на деле является довольно простым. Чтобы запустить любое Windows-приложение достаточно использовать простую команду:

```
wine программа.exe
```

**Опасно:** НИКОГДА НЕ ЗАПУСКАЕТЕ WINE ИЗ ПОД SUDO/ROOT! Это поможет вам избежать проблем в будущем, в том числе с безопасностью.

Немного иной командой запускаются MSI установщики:

```
wine msiexec /i программа.msi
```

При использовании Wine важным понятием является префикс (его также называют бутылкой). Префикс, это как бы файловая система Windows в миниатюре, а по совместительству это рабочая директория, где будут устанавливаться/работать все Windows программы которые вы будете запускать из под Wine. Стоит понимать, что программы запускаемые через Wine по прежнему будут думать что они работают в Windows, хотя на самом деле это не так. Поэтому Wine и понадобилось воссоздать файловую структуру каталогов Windows внутри Linux (Unix). Префикс по умолчанию - это скрытая директория `~/.wine` в папке вашего пользователя. Если вы её откроете то увидите следующее:

```
vasily@archlinux ~> ls ~/.wine
dosdevices  drive_c  system.reg  userdef.reg  user.reg
vasily@archlinux ~> ls -al ~/.wine
итого 3892
drwxr-xr-x 1 vasily users      126 ноя 20 14:53 .
drwx----- 1 vasily users     1188 ноя 20 14:51 ..
drwxr-xr-x 1 vasily users       8 ноя 20 14:51 dosdevices
drwxr-xr-x 1 vasily users     110 ноя 20 14:52 drive_c
-rw-r--r-- 1 vasily users 3646098 ноя 20 14:53 system.reg
-rw-r--r-- 1 vasily users      12 ноя 20 14:51 .update-timestamp
-rw-r--r-- 1 vasily users    4070 ноя 20 14:52 userdef.reg
-rw-r--r-- 1 vasily users  326449 ноя 20 14:53 user.reg
vasily@archlinux ~> 
```

Как мы видим, в префиксе находятся файлы с расширением `.reg` (файлы реестра Windows), директории `dosdevices` и `drive_c`. Файлы реестра используются Wine для, собственно, воссоздания работы реестра Windows в Linux. К ним также будут обращаться программы запускаемые через Wine. Директория `dosdevices` содержит символические ссылки на примонтированные устройства (разделы) в вашей системе Linux. Это понадобилось для того чтобы представить их в виде MS-DOS томов, ибо Windows приложения опять таки не знают что они работают под Linux, и им нужны привычные им диски D, E и т.д. Один из таких "виртуальных дисков" располагается в другом каталоге - `drive_c` (диск C:). Если вы его откроете то увидите "замечательную" структуру каталогов Windows:

```
vasily@archlinux ~/.w/drive_c> cd
vasily@archlinux ~/.w/drive_c> ls -al
итого 0
drwxr-xr-x 1 vasily users 110 ноя 20 14:52 .
drwxr-xr-x 1 vasily users 126 ноя 20 15:00 ..
drwxr-xr-x 1 vasily users   18 ноя 20 14:51 ProgramData
drwxr-xr-x 1 vasily users 118 ноя 20 14:51 'Program Files'
drwxr-xr-x 1 vasily users 118 ноя 20 14:52 'Program Files (x86)'
drwxr-xr-x 1 vasily users   24 ноя 20 14:51 users
drwxr-xr-x 1 vasily users  492 ноя 20 14:52 windows
vasily@archlinux ~/.w/drive_c> 
```

Именно сюда и будут устанавливаться все Windows программы и работать они как правило тоже будут именно там.

Вы можете переназначить префикс по умолчанию через переменную окружения `WINEPREFIX`, указав Wine использовать другую директорию для его расположения вместо `~/.wine`. Например:

```
WINEPREFIX=~/Games wine game.exe # Если директории не было, он её создаст.
```

Понятное дело, что при смене префикса через переменную окружения `WINEPREFIX` не переносится его содержимое, т.е. программы установленные в одном префиксе не будут скопированы в новый. Но если вам нужно просто сменить название префикса с сохранением его содержимого, то просто переименуйте название директории, а затем переназначьте переменную, например:

```
mv ~/old_wineprefix ~/new_wineprefix
WINEPREFIX=~/new_wineprefix wine приложение.exe
```

Префиксы бывают 32-битные и 64-битные в соответствии с разрядностью систем Windows (по умолчанию создаются 64-битные). Указать разрядность префикса можно через переменную `WINEARCH`. Для запуска старых видеоигр мы рекомендуем использовать 32-битный префикс во избежание проблем с совместимостью:

```
WINEPREFIX=~/wine32 WINEARCH=win32 wine oldgame.exe
```

Если вы уже создали 64-битный префикс, то переназначить его разрядность через переменную `WINEARCH` не получится. Создайте новый и перенесите нужную вам программу.

Проверить разрядность уже существующего префикса можно командой (можно также проверить по наличию директории "Program Files (x86)" внутри префикса):

```
grep '#arch' ~/.wine/system.reg
```

(Где '.wine' - путь до нужного вам префикса)

## 8.1.5 DXVK

В Linux отсутствует полноценная реализация DirectX по вполне понятным причинам. Но присутствуют альтернативные графические API, работающие под любые платформы. Прежде всего это OpenGL и Vulkan. В следствии этого в Wine есть так называемый ретранслятор кода - wined3d. Он переводит вызовы DirectX в известные любой Linux системе OpenGL вызовы. Однако OpenGL не одно и тоже что и DirectX, поэтому возникают множественные проблемы. Самая главная из которых - значительно более худшая производительность OpenGL по сравнению с DirectX. Именно поэтому если вы запустите любую игру через "голый" Wine вы получите ужасный FPS, т.к. она будет работать через wined3d. По этой причине был разработан другой ретранслятор кода - DXVK. Он переводит DirectX вызовы уже не в OpenGL, а в Vulkan - более современный графический API, который достигает паритета по своим возможностям и производительности с DirectX.

Установка DXVK - это первое что должен сделать любой игрок который собирается запустить Windows-игру под Linux. Но для любой версии Proton DXVK уже есть из коробки, а вот для Wine его придется действительно устанавливать вручную.

Мы рекомендуем собирать `dxvk-mingw` из GitHub для лучшей производительности и активации асинхронного патча. Если чуть подробнее, то данный патч позволяет выполнять компиляцию шейдеров в асинхронных потоках. Такой подход позволяет минимизировать заикания во время игры, которые могут происходить когда вы прогружаете новую локацию или объект на игровой карте, то есть компилируйте новые шейдеры. В некоторых

играх он даже немного повышает FPS и делает график времени кадра более плавным. Патч не был одобрен разработчиками потому, что он потенциально вызывал проблемы в онлайн-играх с анти-чит системами, и теперь для него требуется отдельная установка.

### Установка::

```
git clone https://github.com/loathingKernel/PKGBUILDs
cd PKGBUILDs/public/dxvk-mingw
mv PKGBUILD.testing PKGBUILD
sed -i 's/patch -p1 -i "$srcdir"/\1582\.patch//g' PKGBUILD
sed -i 's/patch -p1 -i "$srcdir"/\1582-fix-include\.patch//g' PKGBUILD
sed -i 's/-O3 -march=haswell -mtune=haswell -pipe/-O2 -march=native -mtune=native -
    pipe/g' PKGBUILD # Нативные флаги
makepkg -sruc # Сборка и установка
```

Активировать асинхронную компиляцию шейдеров можно через переменную окружения `DXVK_ASYNC=1`.

После установки пакета DXVK не задействуется сразу, его библиотеки ещё нужно "распаковать" по отдельности в каждый префикс Wine (это не относится к играм запускаемым через Lutris/Proton, в них DXVK включён по умолчанию):

```
WINEPREFIX=~/prefix setup_dxvk install # Где "prefix" - это путь до вашего префикса
    Wine
```

**Предупреждение:** DXVK осуществляет ретрансляцию вызовов только для игр использующих версии DirectX 9, 10 и 11. Для DirectX 12 для понадобиться использовать vkd3d. Подробнее о нем вы можете прочитать ниже.

**Опасно:** С осторожностью используйте `DXVK_ASYNC=1` в онлайн-играх!

## 8.1.6 vkd3d

vkd3d - это ретранслятор кода, аналогичный DXVK, но уже конкретно для версии DirectX 12. Стоит отметить, что существует две отдельно разрабатываемые версии vkd3d, одна из которых разрабатывается командой Wine, а другая - Valve. Мы рекомендуем вам использовать ту что от Valve, т.к. она наиболее заточена под современные игры, а также достаточно хорошо поддерживает Raytracing.

### Установка vkd3d-proton

Для Proton и Lutris установка vkd3d задействован по умолчанию, и никаких дополнительных манипуляций обычно не требуется. Однако для обычного Wine нужна его отдельная установка. Мы установим vkd3d-proton из AUR, нативно-скомпилировав его под свой процессор:

```
git clone https://aur.archlinux.org/vkd3d-proton-mingw.git # Скачивание исходников
cd vkd3d-proton-mingw                                # Переход в директорию
sed -i 's/-O3 -march=nocona -mtune=core-avx2 -pipe/-O2 -march=native -mtune=native -
    pipe/g' PKGBUILD # Нативные флаги
makepkg -sruc                                         # Сборка и установка
```

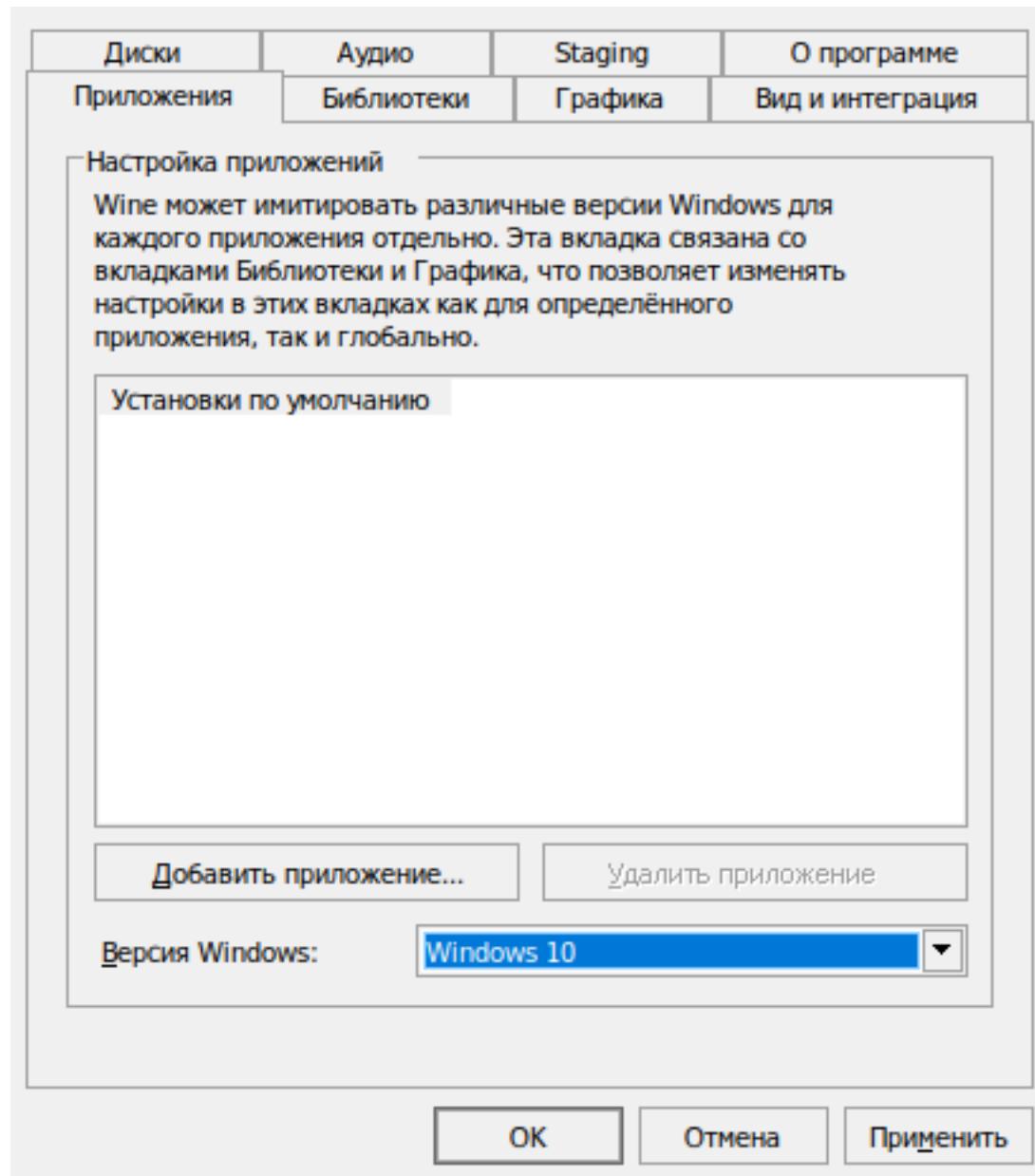
Так же как и в случае с DXVK, после установки пакета, vkd3d нужно предварительно распоковать в нужный Wine префикс:

```
setup_vkd3d_proton install ~/.wineprefix
```

(Где '~/.wineprefix' - это путь до нужного вам префикса)

Кроме того, обязательно измените версию Windows вашего префикса на "Windows 10":

```
WINEPREFIX=~/wineprefix winecfg
```



## 8.1.7 Полезные ссылки по теме Wine и DXVK

### Видео настройке Бинарной версии Wine.

<https://www.youtube.com/watch?v=NKI3dtK7mRI> (Устаревшее видео).

### Скачать готовые сборки Wine и DXVK

<https://mega.nz/folder/pNsTiQyA#2vur9shHbXvLnhdQTpd3AQ>

<https://mega.nz/folder/IJdEgIrT#wXcbgymIDP2mesJ8kE99Qg>

<https://github.com/Kron4ek/Wine-Builds>

<https://mirror.cachyos.org/?search=wine>

### Почитать, что это такое

<https://www.newalive.net/234-sborki-dxvk-i-d9vk.html>

<https://www.newalive.net/231-wine-tk-glitch.html>

## 8.2 Дополнительные компоненты

Не являются обязательными, но могут помочь повысить производительность системы или облегчить настройку.

### 8.2.1 Lutris

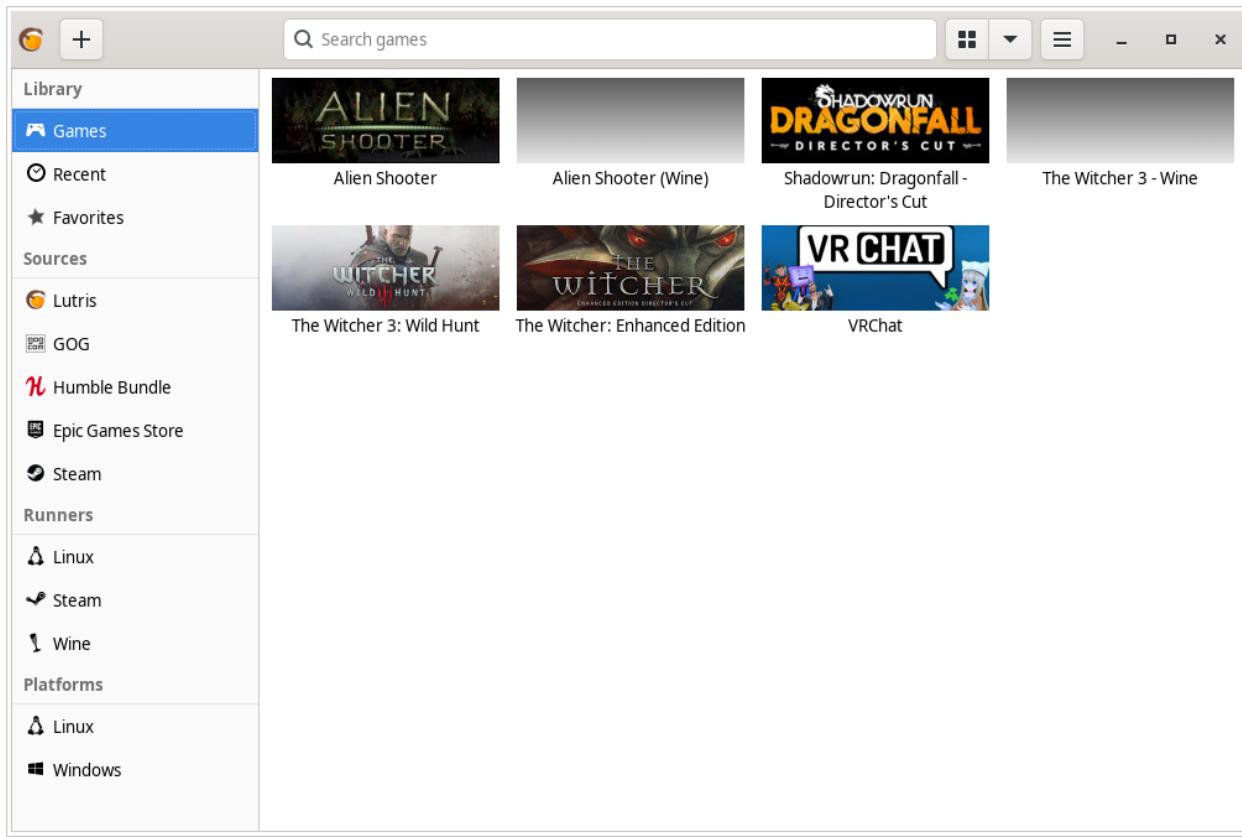
Lutris - это удобный графический интерфейс по обслуживанию всей вашей игровой библиотеки (включая все купленные игры Steam/GOG/Epic Games) в одном приложении. Через него вы сможете достаточно просто запускать нативные игры, игры запускаемые при помощи эмуляторов, и конечно Wine. Все это объединено в одном приложении-комбайне, содержащим много настроек и интеграций с различными сервисами.

#### Установка

Все проще некуда:

```
sudo pacman -S lutris
```

Тем не менее, стоит удостовериться что вы установили полный набор зависимостей для Wine. Об этом вы можете прочитать в предыдущих разделах.

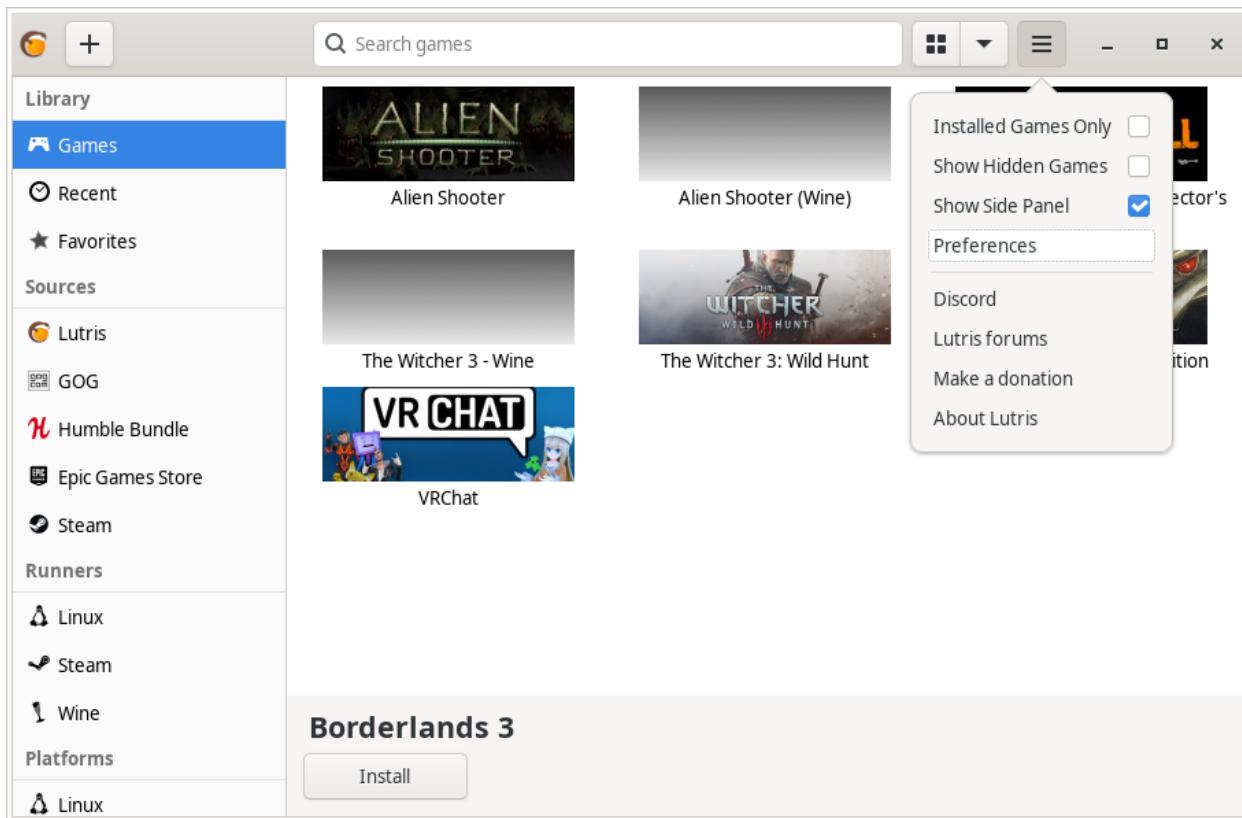


## Интеграция с GOG/Epic/Steam

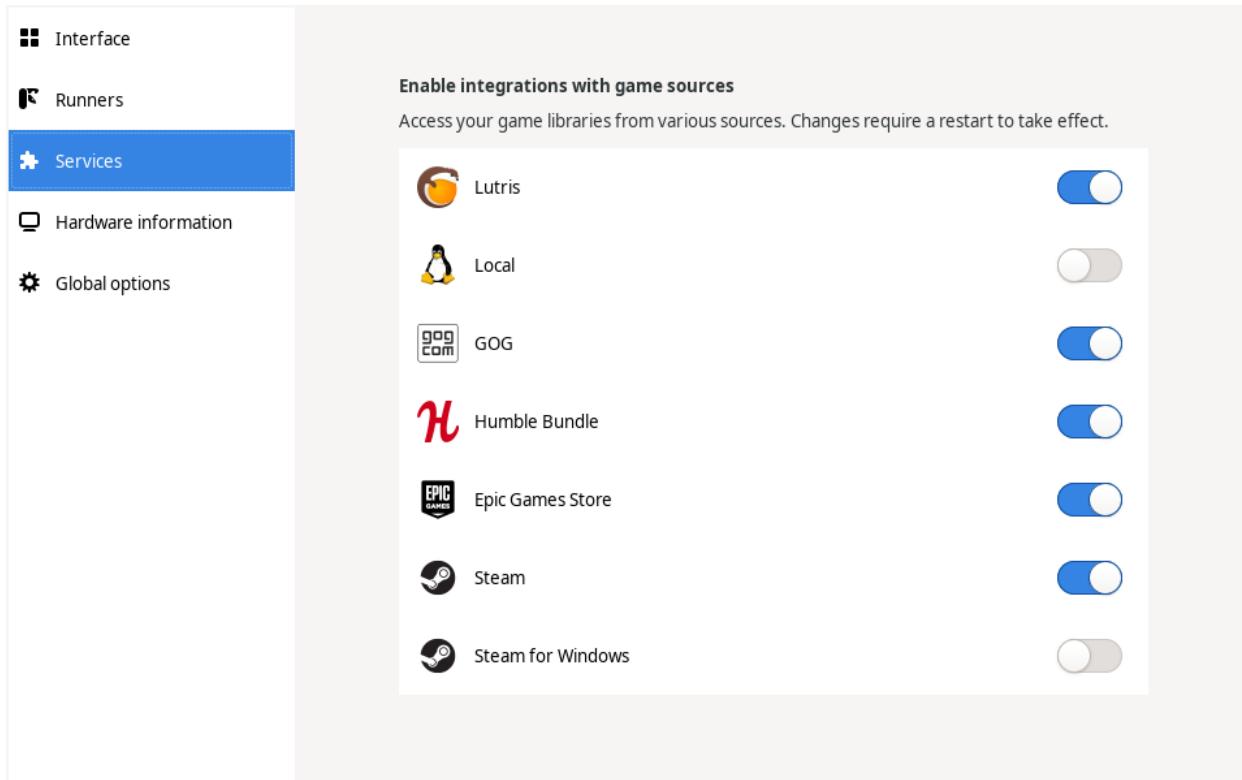
Сразу после установки стоит сделать некоторые базовые вещи. А именно подключить интеграцию с сервисами Steam/GOG/Epic Games. Это позволит синхронизировать локальную библиотеку Lutris'a вместе с перечисленными площадками и выполнять установку игр в два клика. Подключать все конечно не обязательно, так что делайте это если считаете нужным.

**1.** Зайдем в настройки: В правом верхнем углу найдите три горизонтальные полоски и в контекстном меню выберите "Preferences". После этого выберите "Services" и включите те сервисы, которыми вы пользуетесь.

### 1.1



### 1.2

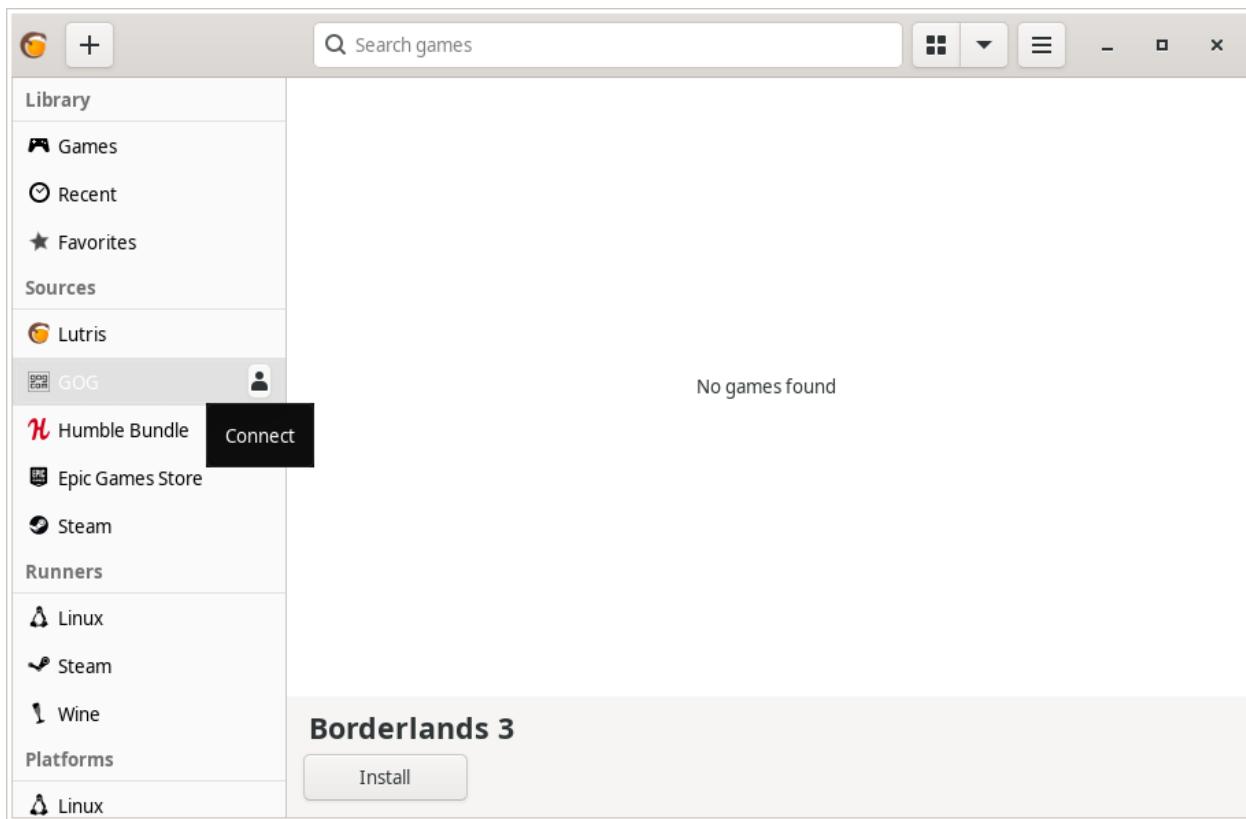


2. Теперь вернитесь в главное окно и наведите курсор на левую панель в графу "Sources",

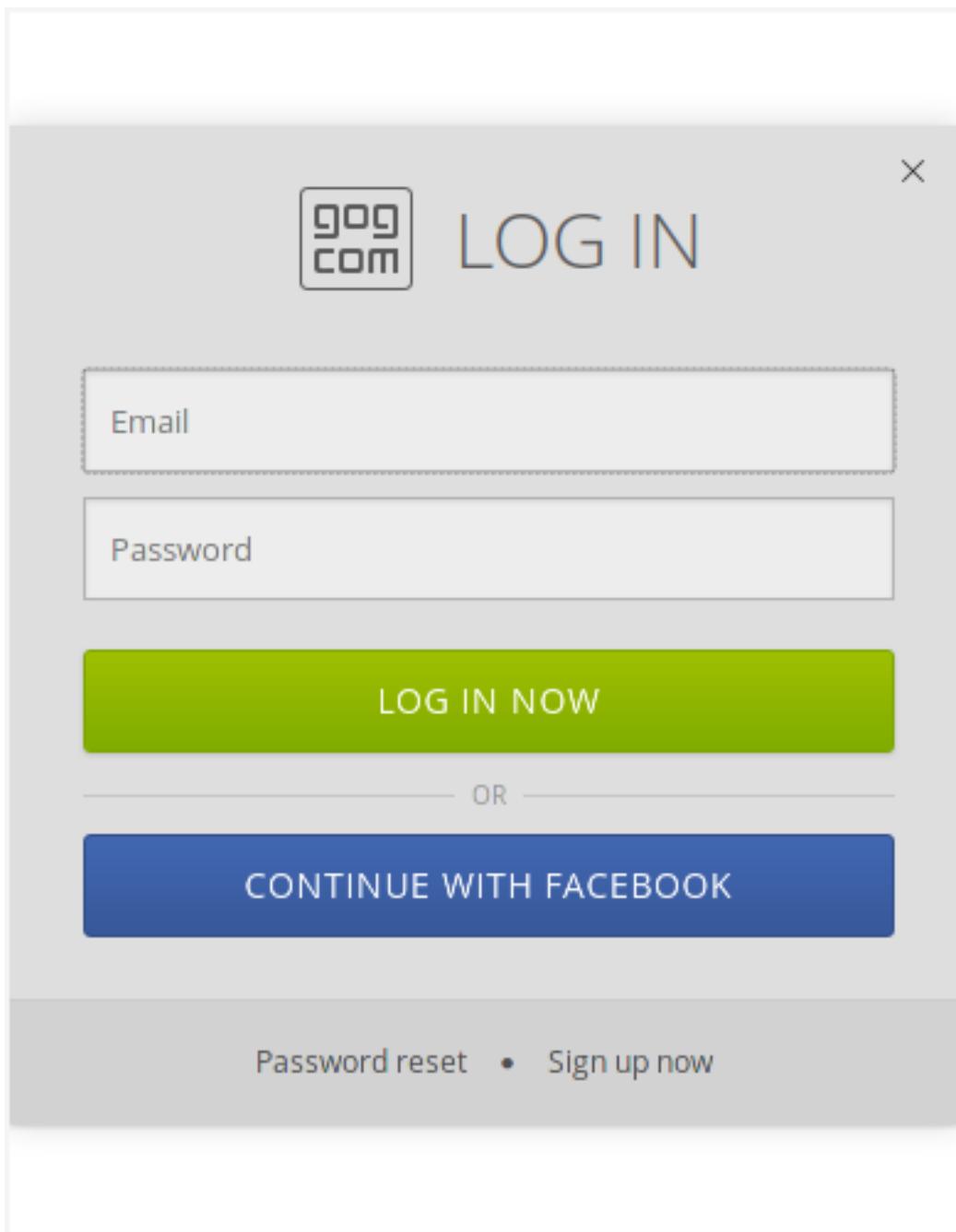
и ниже выберите нужную вам платформу. Справа от курсора будет иконка входа. После этого перед вами появится окно авторизации, после прохождения которой у вас появится возможность устанавливать и запускать все игры из вашей внешней библиотеки (Steam/GOG/Epic Games).

Пример подключения аккаунта GOG представлен ниже на скриншотах.

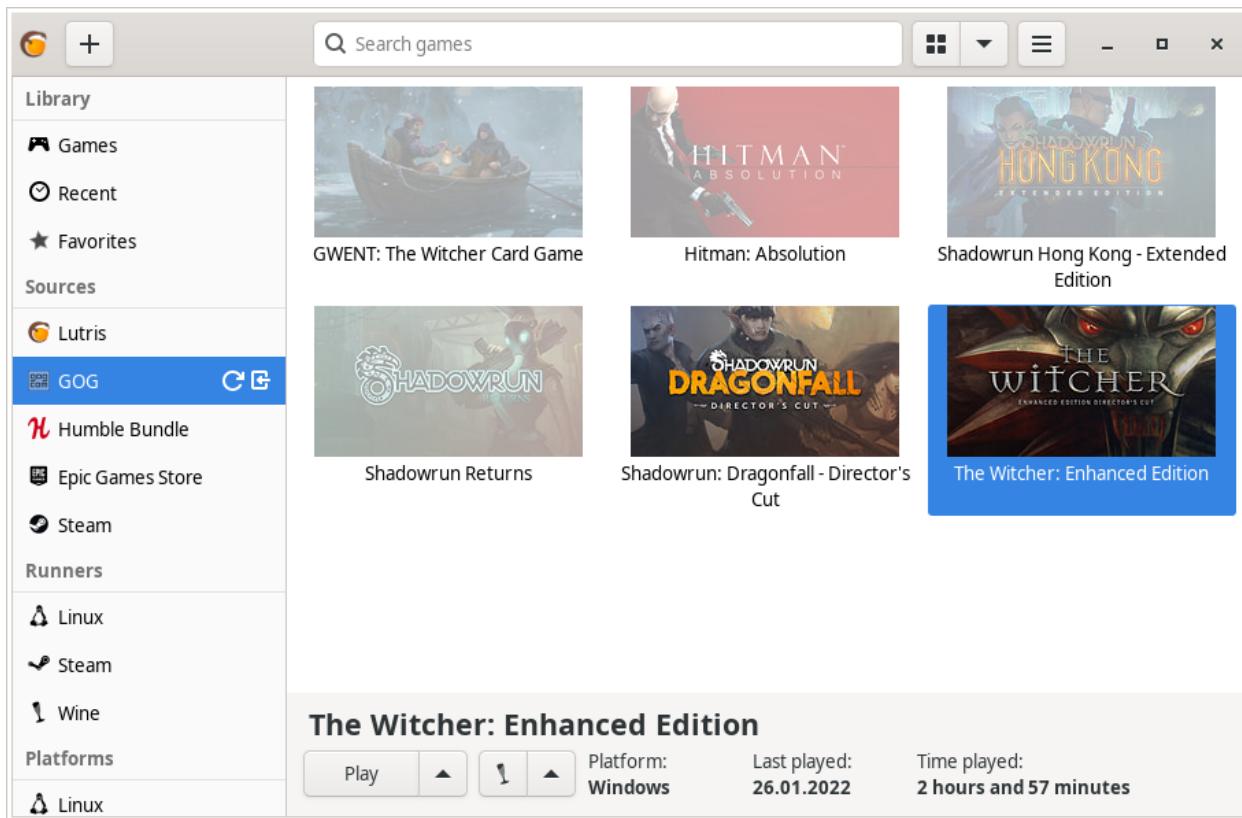
## 2.1



## 2.2

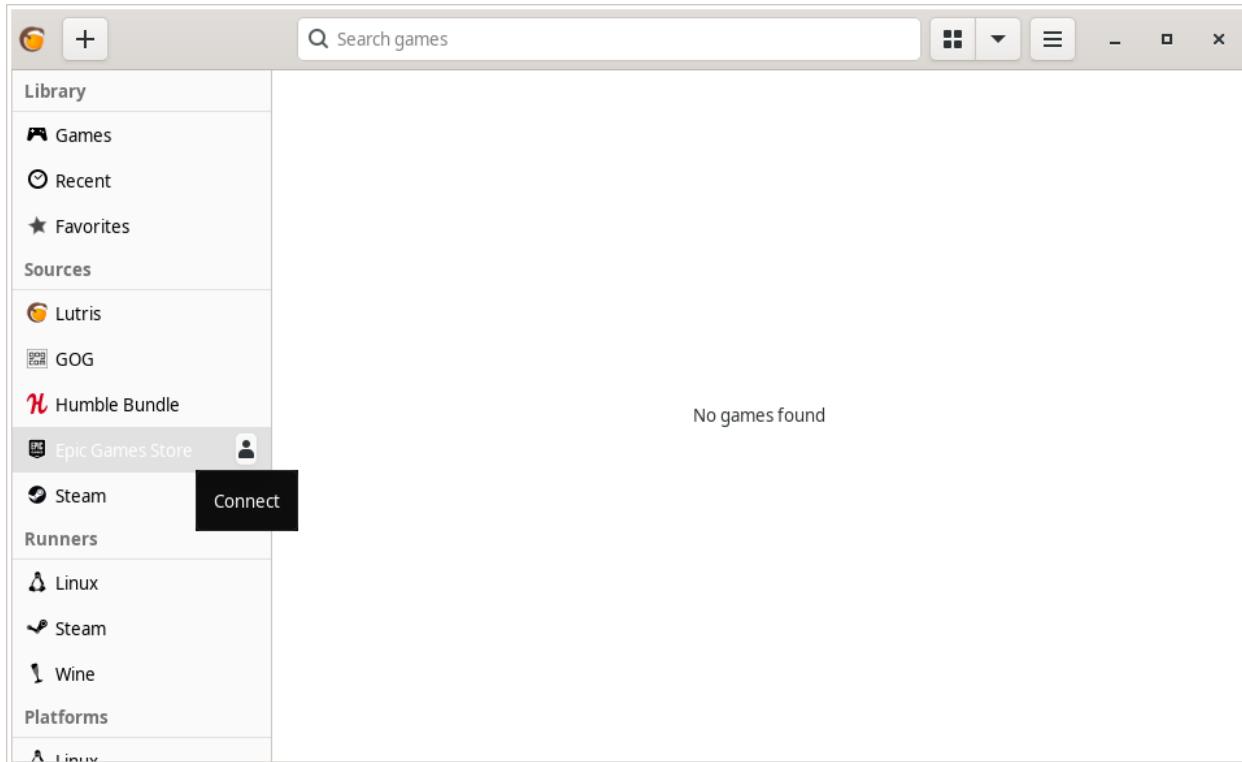


2.3

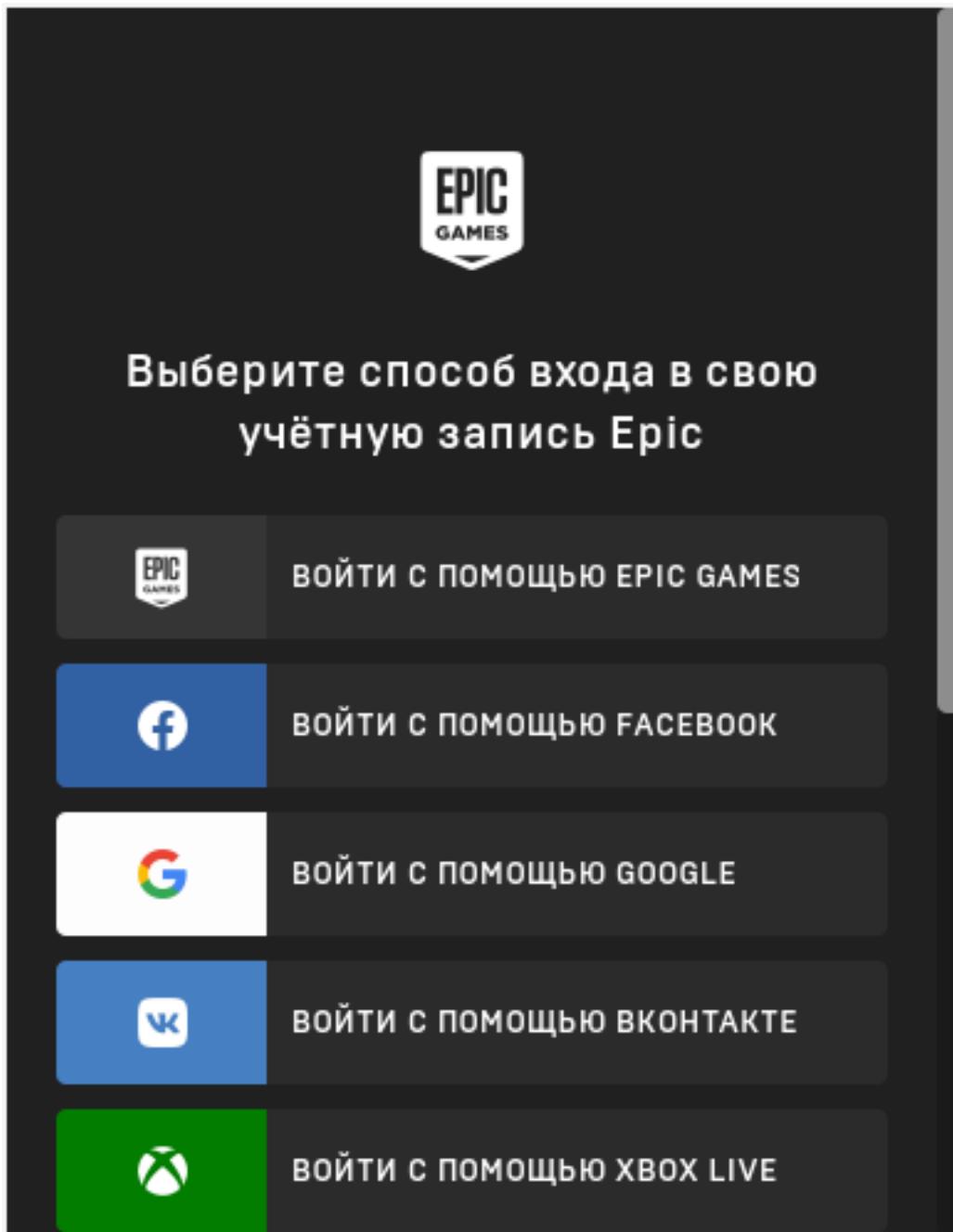


Аналогичная операция проделывается с Epic Games Store:

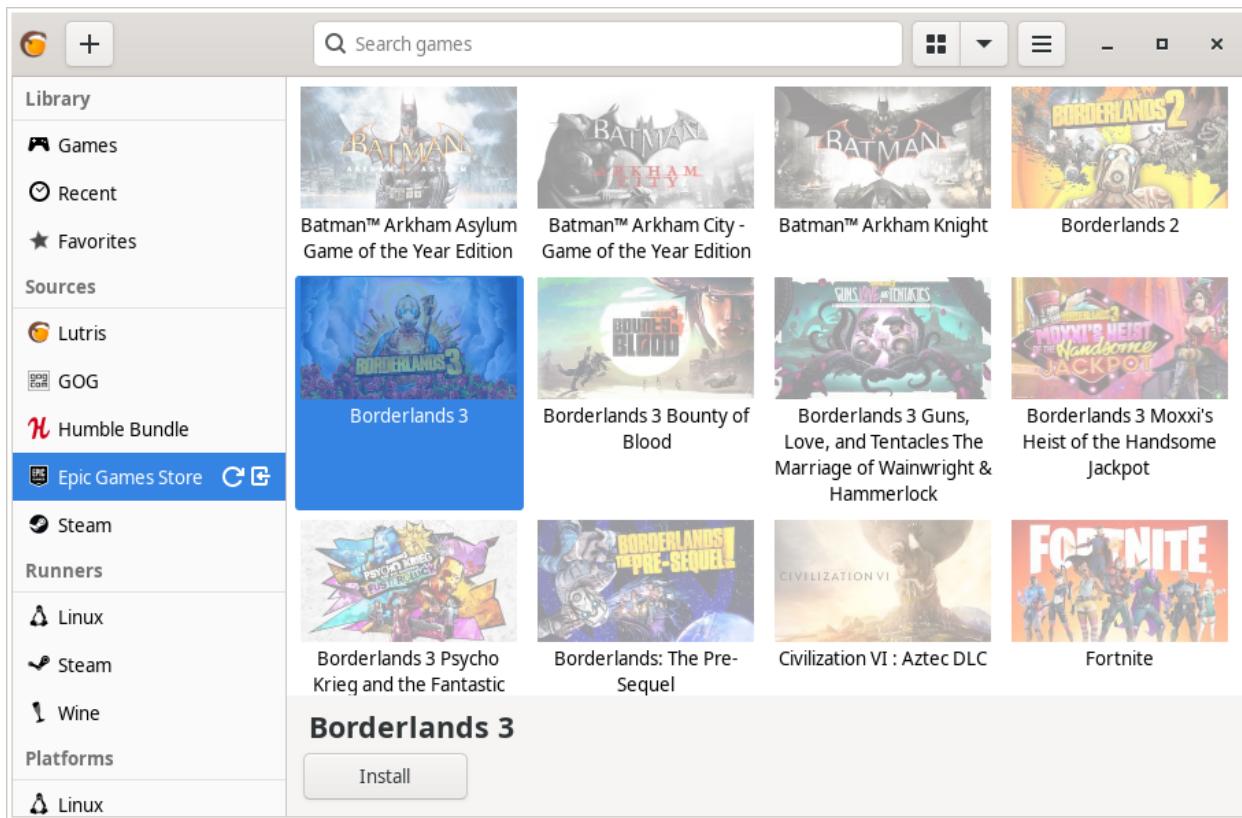
## 2.4



2.5



2.6



## Пример работы с Lutris

<https://www.youtube.com/watch?v=ybe0MzJDUvw>

### 8.2.1.1 Использование Proton-GE-Custom в Lutris

Немногие понимают, что Proton по сути является тем же Wine, хоть и с плюшками. Так вот, зная этот факт, мы можем сказать Lutris использовать Proton в качестве кастомного Wine. Делается это очень просто:

```
mkdir -p ~/.local/share/lutris/runners/wine
ln -s /usr/share/steam/compatibilitytools.d/proton-ge-custom/files ~/.local/share/
↪ lutris/runners/wine/wine-proton-ge
```

Затем просто выберите пункт в выборе версии Wine на "wine-proton-ge" в Lutris для нужной вам игры.

## 8.2.2 Gamemode

Gamemode - утилита для максимальной выжимки системы во время игры. Установку gamemode можно выполнить следующей командой:

```
sudo pacman -S gamemode lib32-gamemode
```

Lutris, как правило использует gamemode по умолчанию (в случае его наличия в системе), однако вы также можете активировать или деактивировать его в параметрах.

Для запуска игры вручную с использованием gamemode необходимо выполнить комманду:

```
gamemoderun ./game
```

Для запуска игр через Steam с использованием gamemode необходимо прописать комманду в параметрах запуска игры (находятся в свойствах игры в Steam):

```
gamemoderun %command%
```

## 8.2.3 AMD FidelityFX Super Resolution в Wine

Возможно, вы слышали о волшебной технологии DLSS от Nvidia, которая позволяет поднять FPS почти в два раза и при этом не потратить ни копейки на новое оборудование. Вот и компания AMD совсем недавно представила похожую технологию, которая получила помпезное название AMD FidelityFX Super Resolution или сокращенно FSR. Новая технология масштабирования картинки от AMD не требует наличия дорогой карты или каких-то аппаратных блоков ускорения, что в отличие от DLSS, должно позволить использовать технологию везде и совершенно бесплатно. А благодаря чудесным патчам от энтузиастов для Wine мы можем применять эту волшебную технологию для любой Windows-игры.

### I. Установка

Чтобы установить патч от энтузиастов придется немножко помудрить с нашим wine-tkg.

Его установка описывалась выше, но чтобы задействовать сторонний патч на FSR в Wine нужно отредактировать одну строку в *customization.cfg*:

```
nano customization.cfg

# Найдите строчку _community_patches="" и добавьте в неё следующее:

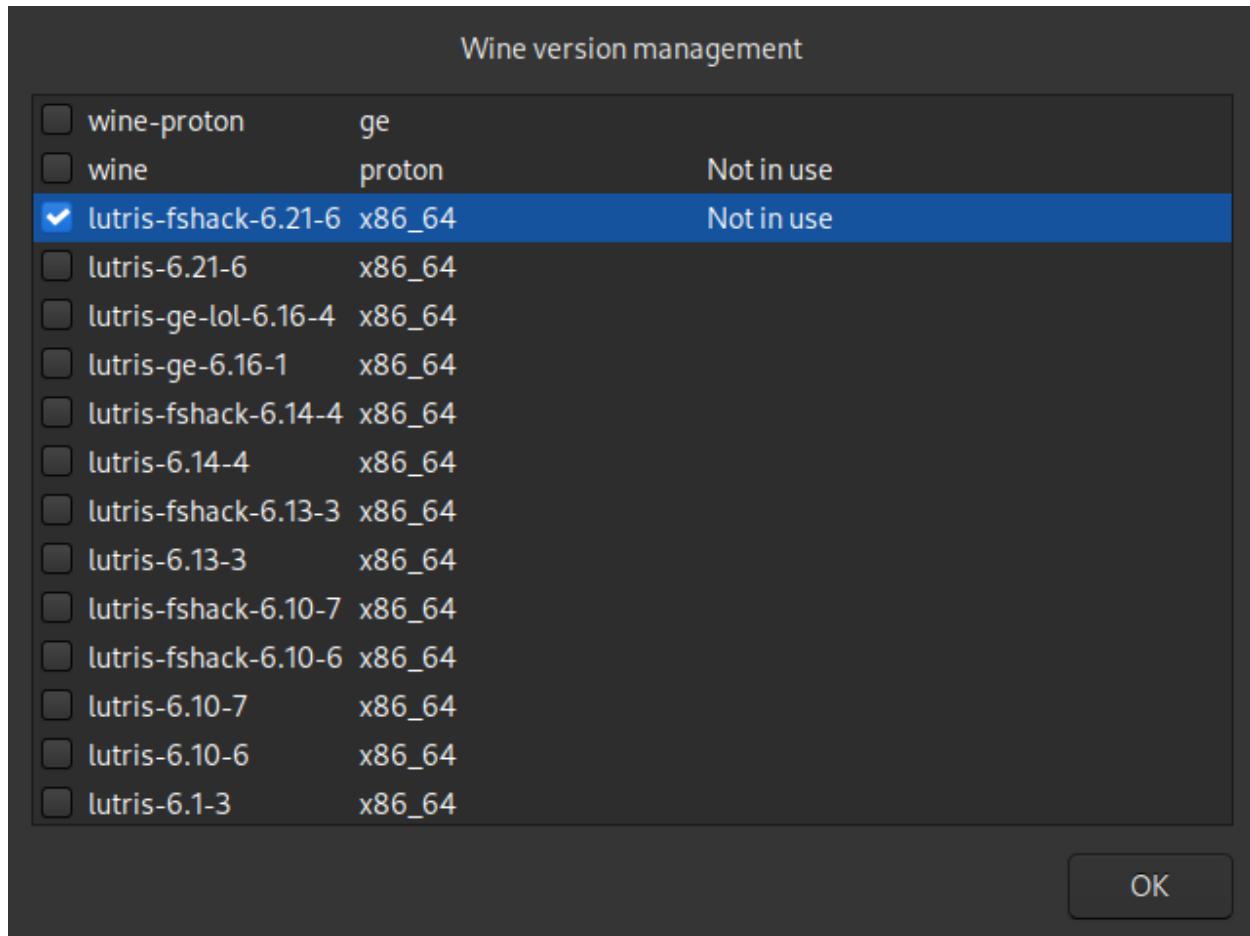
_community_patches="amd_fsr_fshack.mypatch"

# Обязательно оставьте при этом включенными данные параметры:
_protonify, _msvcrt_nativebuiltin, _proton_fs_hack, _proton_rawinput.
Без них ничего работать не будет.
```

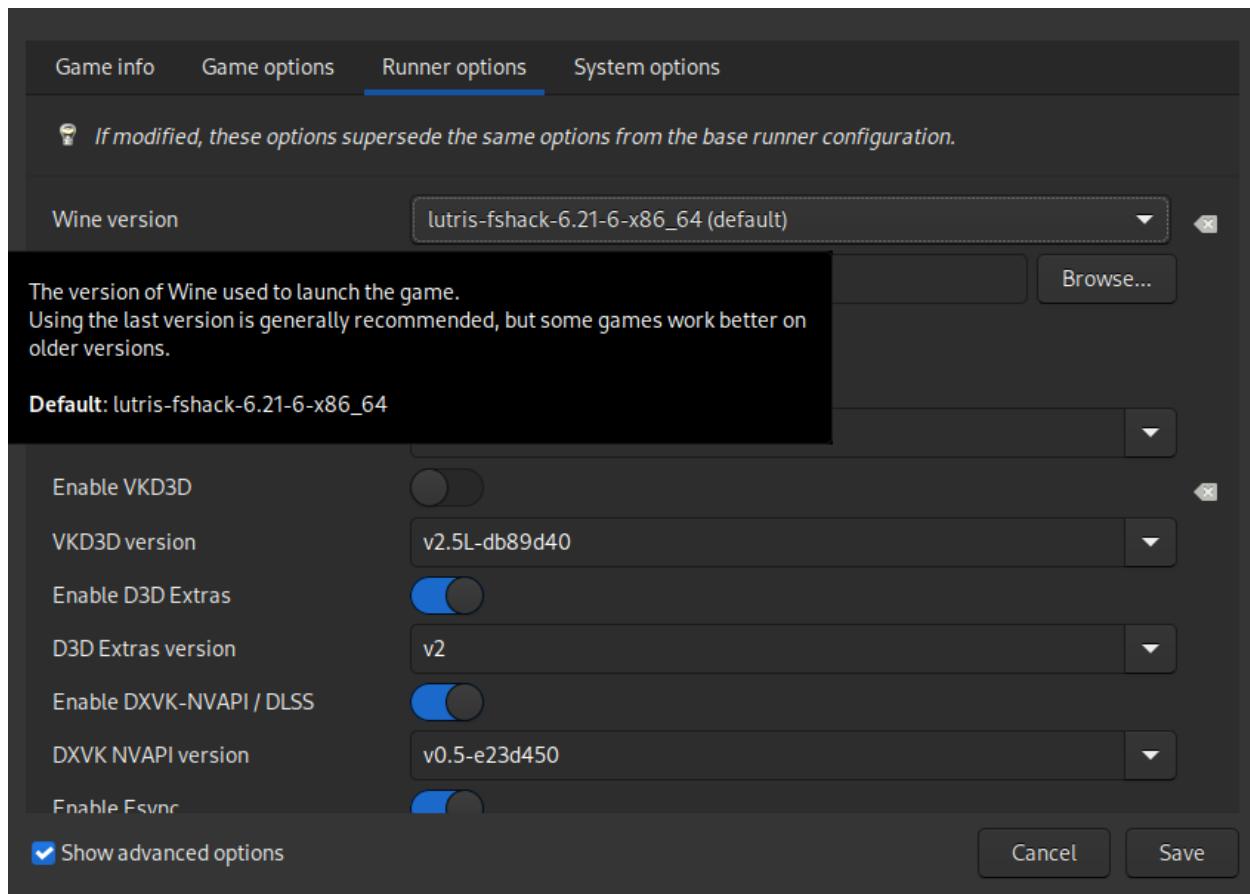
И пересоберите ваш wine-tkg: `makepkg -sric`

### II. Установка

Если вам кажется первый способ немного муторным, то вы можете просто использовать уже готовые сборки с FSR патчем в Lutris:



И затем выбрать её для нужной вам игры:



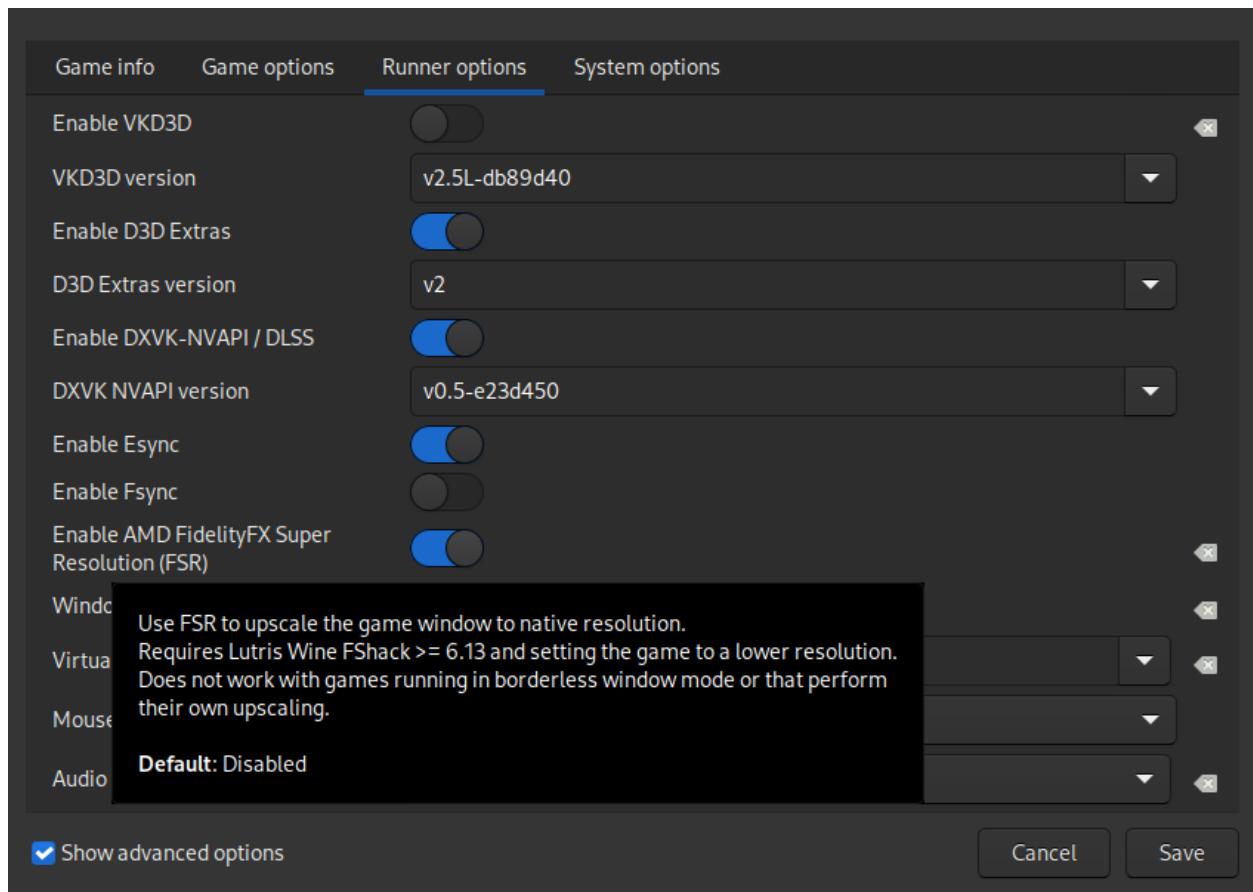
### III. Установка

FSR патч также по умолчанию задействован в Proton-GE-Custom. Про его установку вы можете прочитать ниже в соответствующем разделе.

#### Как использовать

Несмотря на то, что мы выполнили установку патченной версии Wine одним из вышеописанных способов, технологию FSR ещё нужно активировать.

Сделать это можно руками, через переменные окружения `WINE_FULLSCREEN_FSR=1` или в Lutris:



Важно помнить, что эта технология работает **только в полноэкранном режиме игры**.

Регулировать резкость итогового изображения можно через переменную окружения `WINE_FULLSCREEN_FSR_STRENGTH=N`, где N - это уровень резкости изображения от 0 до 5. Чем выше значение, тем меньше резкость. По умолчанию установлено значение "2", мы рекомендуем использовать значение "3".

#### Видеоверсия и демонстрация работы технологии

<https://www.youtube.com/watch?v=YNhwAazJODU>

#### 8.2.4 Использование DLSS с видеокартами NVIDIA через Proton

Для того чтобы использовать DLSS вам потребуется:

- Видеокарта поддерживающая данную технологию (видеокарты серии RTX и выше).
- Убедиться, что используемая версия Proton не ниже **6.3-8!** (**поддержка DLSS начинается с данной версии!**)
- Указать параметры запуска игры в свойствах игры Steam `PROTON_HIDE_NVIDIA_GPU=0 PROTON_ENABLE_NVAPI=1`
- Некоторые игры, как правило, которые используют DX11, для корректной работы могут также потребовать включения `dxgi.nvapiHack = False` в `dxvk.conf`. Для этого выполните инструкции ниже:

```
mkdir -p ~/.config/dxvk/dxvk.conf  
echo "dxgi.nvapiHack = False" > ~/.config/dxvk/dxvk.conf
```

После этого не забудьте дописать *DXVK\_CONFIG\_FILE=~/config/dxvk/dxvk.conf* в приведённом ниже примере перед %command%.

Пример для использования в Steam:

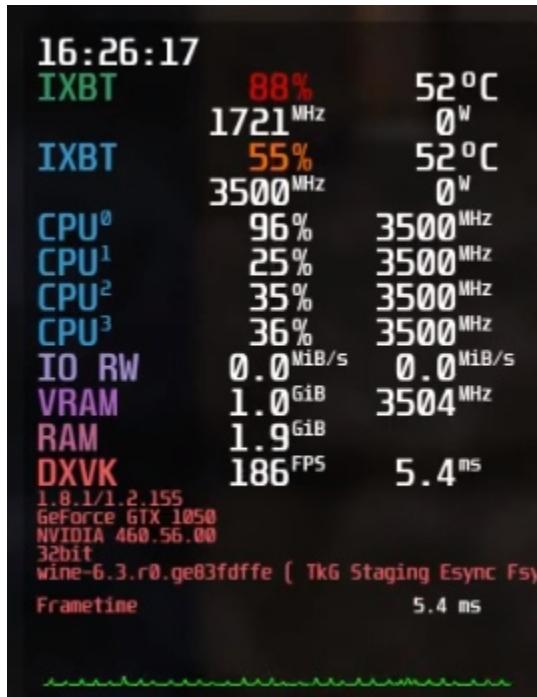
```
PROTON_HIDE_NVIDIA_GPU=0 PROTON_ENABLE_NVAPI=1 %command%
```

**Внимание:** Поскольку для DLSS необходимо специальное машинное обучение, то для запуска необходимо чтобы игра поддерживала DLSS, т.е. в настройках игры должен быть параметр включения данной функции. **Иначе DLSS работать не будет!**

## 8.2.5 Мониторинг FPS в играх.

### 8.2.5.1 Mangohud

Включение мониторинга в играх как в MSI Afterburner.



### Установка

```
cd tools  
→ папку в домашнем каталоге.  
git clone https://aur.archlinux.org/mangohud.git  
cd mangohud  
makepkg -srcc # Переход в заранее созданную.  
# Скачивание исходников.  
# Переход в mangohud.  
# Сборка и установка.
```

Графический помощник для настройки вашего Mangohud.

```
cd tools                                # Переход в заранее созданную папку
→ В домашнем каталоге.
git clone https://aur.archlinux.org/goverlay.git # Скачивание исходников.
cd goverlay                               # Переход в goverlay-bin
makepkg -sr                   # Сборка и установка.
```

Для использования mangohud в играх через Steam необходимо добавить команду в параметры запуска игры (находятся в свойствах игры Steam):

```
mangohud %command%
```

(Для указания нескольких команд необходимо разделять их **пробелом**)

**Подробней в видео.**

<https://www.youtube.com/watch?v=4RqerevPD4I>

#### **8.2.5.2 Альтернатива: DXVK Hud (Только для игр запускаемых через Wine/Proton)**

Вы также можете использовать встроенную в DXVK альтернативу для мониторинга - DXVK Hud. Он не такой гибкий как MangoHud, но также способен выводить значения FPS, график времени кадра, нагрузку на GPU. Использовать данный HUD можно задав переменную окружения *DXVK\_HUD*. К примеру, *DXVK\_HUD=fps,frametimes,gpupload* выводит информацию о FPS, времени кадра, и нагрузке на GPU.

Полный список значений переменной вы можете узнать - [здесь](#).

# ГЛАВА 9

## Сборка мини-ядра, и с чем это едят.

Ядра, что мы скомпилировали выше уже дают существенное повышение производительности системы, однако мы еще выжали не все соки. По умолчанию ядра собираются для универсального применения на разном оборудовании, т.е. с наличием различных модулей и драйверов для всякого рода периферии и железа, которого у вас могло никогда и не быть.

**Мини-ядро** - Это Linux ядро собранное с минимальным количеством модулей/драйверов необходимых для работоспособности вашего железа.

Плюсы: Значительное сокращение времени на сборку ядра, уменьшение размера ядра, повышение производительности.

Минусы: Невозможность использования нового оборудования или портов без повторной пересборки ядра.

Чтобы собрать мини-ядро, нам нужно:

Установить modprobed-db по аналогии с другими AUR пакетами.

После установки выполнить:

```
systemctl --user enable --now modprobed-db.service # Это демон для индексирования  
↪активно используемых системой модулей ядра  
sudo modprobed-db recall # Сделает дамп используемых системой модулей ядра.
```

Далее, активно используем всю периферию и железки, что у вас есть пока не соберется достаточно большое количество модулей (Примерно 2-3 дня активного пользования системой).

После того как все приготовления сделаны, просто собираем ядро как было указано выше, но перед сборкой (*makepkg -si*) нужно отредактировать PKGBUILD:

```
nano PKGBUILD
```

И меняем значение этой строки (работает почти для любых ядер): *\_localmodcfg=y*

Все, теперь собираем мини-ядро по аналогии с обычным.

*P.S.* Если при сборке образов уже скомпилированного ядра выдает ошибку с указанием на отсутствующие модули, что-то в формате: db\_xxx, bd\_xxx - просто пропишите их в ручную:

```
sudo nano ~/.config/modprobed.db
```

Затем выполните:

```
sudo modprobed-db store  
sudo modprobed-db recall
```

И снова пересоберите ядро.

#### **Видео версия**

<https://www.youtube.com/watch?v=8GRNN94afyg>

## **9.1 Возможные часто встречающиеся проблемы после установки мини-ядра**

**П:** Система не загружается дальше rootfs (частая проблема).

**Р:** Обычно это означает, что какие-то системно-важные модули не были "подхвачены" modprobed-db. Почти всегда дело заключается в модулях на поддержку SATA/SCSI, либо ATA и модулей Файловых систем.

Вот список модулей, из-за отсутствия которых может не грузиться система:

- scsi\_mod
- sd\_mod
- libahci
- libata
- lzo\_rle
- efi\_pstore
- evdev
- ext4
- btrfs
- ahci
- autofs4
- fuse
- dm\_cache
- dm\_cache\_smq
- dm\_mirror
- dm\_mod
- dm\_snapshot
- dm\_thin\_pool

Чтобы это исправить просто добавьте эти модули вручную, т.е. отредактировав файл по пути `sudo nano ~/.config/modprobed.db`. Затем снова пересоберите мини-ядро как это показано в предыдущем разделе, после пересборки мини-ядро должно загрузиться.

**П:** После установки мини-ядра отсутствует интернет-подключение.

**Р:** Обычно это вызвано отсутствием модулей драйвера для сетевой карты, либо отсутствием важных системных модулей для корректной работы интернет подключения. Вот список модулей, из-за которых возможно не работает сеть:

- 8021q
- af\_packet
- af\_alg
- alx
- ecdh\_generic
- garp
- libphy
- r8169
- rc\_core
- realtek
- sch\_fq\_codel
- llc

Так же, как и в случае с прошлой проблемой, просто пропишите эти модули в ручную, т.е. отредактируйте `sudo nano ~/.config/modprobed.db`. Обратите внимание, что модуль драйвера для сетевой карты у каждого может быть разный, и перед тем как прописать какой-либо модуль драйвера, лучше посмотреть в рабочей системе (`lspci -v`) какой именно нужен вашей сетевой карте, и прописать его. После этого, в очередной раз, пересоберите мини-ядро.

**П:** После перезагрузки драйвер NVIDIA загружается, но вместо него используется llvmpipe.

**Р:** Укажите точный путь до модулей драйвера в ваших настройках Xorg, т.е. пропишите в `/etc/X11/xorg.conf` следующее:

```
Section "Files"
    ModulePath "/usr/lib/nvidia/xorg"
    ModulePath "/usr/lib/xorg/modules"
EndSection
```

Затем перезагрузитесь.

# ГЛАВА 10

---

## Оптимизация рабочего окружения (DE)

---

Современные среды рабочего стола стали достаточно прожорливыми и требовательными к аппаратным ресурсам компьютера, и хотя они и довольно хороши с точки зрения удобства использования, все же хотелось бы минимизировать потребление аппаратных ресурсов с их стороны. Поэтому в этом разделе мы по отдельности и рассмотрим оптимизацию разных рабочих окружений и не только.

### 10.1 Запуск любой DE или WM без экранного менеджера (Только для X11)

Почти всегда любое рабочее окружение запускается при помощи экранного менеджера (его ещё называют менеджером входа), через который вы осуществляете вход в систему и оболочку соответственно. Говоря ещё проще, это экран входа, где вас просят пройти аутентификацию (ввести пароль к вашей учетной записи). Он также выполняет функцию управления рабочими сессиями в разных окружениях. Тем не менее он тоже потребляет определенные ресурсы компьютера, а осуществить вход в систему можно и без него, т.е. через tty, хоть вы и пожертвуете тем самым определенным уровнем удобства. Для автоматизации запуска любой DE/WM (кроме Wayland сессий) через tty вам понадобиться прописать в ваш `.text_profile` или `.zsh_profile` следующее:

```
if [[ -z $DISPLAY && $(tty) == /dev/tty1 ]]; then
    XDG_SESSION_TYPE=x11 GDK_BACKEND=x11 exec startx
fi
```

Это запустит X-сервер сразу при входе в tty1 (терминал по умолчанию). Обязательным условием при этом является наличие установленного пакета `xorg-xinit`, и заранее настроенный файл `~/.xinitrc`, в котором прописана команда запуска вашего DE/WM. Например: `exec gnome-session #` Запускает gnome сессию при запуске Xorg сервера.

## 10.2 GNOME 3.XX/42

Сам по себе GNOME - наверное одна из самых тяжеловесных и требовательных к аппаратным ресурсам оболочки из ныне существующих. Тем не менее у неё есть свои преимущества перед другими оболочками, за что её и любят пользователи. Но к сожалению низкое энергопотребление не в их числе, поэтому в этом разделе вы узнаете о том, как заставить похудеть ваш толстенький gnome-shell.

### 10.2.1 Удаление мусора GNOME

```
sudo pacman -Rsn epiphany gnome-books gnome-boxes gnome-calculator gnome-calendar  
→ gnome-contacts gnome-maps gnome-music gnome-weather gnome-clocks gnome-photos gnome-  
→ software gnome-user-docs totem yelp gvfs-afc gvfs-goa gvfs-gphoto2 gvfs-ftp gvfs-  
→ hfs gvfs-smb gvfs-google vino gnome-user-share gnome-characters simple-scan eog  
→ tracker3-miners rygel nautilus evolution-data-server gnome-font-viewer gnome-remote-  
→ desktop gnome-logs orca
```

**P.S.** Удаляйте пакеты с осознанием того, что вы делаете. Несмотря на то, что здесь были собраны наиболее сомнительные по соотношению нужности/прожорливости пакеты, вы можете найти какой-либо из пакетов полезным и нужным.

**Предупреждение:** Некоторые пакеты из вышеприведенной команды могут быть не найдены в вашей системе. В таком случае просто выпишите их из команды.

Для совсем отчаянных парней, после окончательной настройки параметров GNOME, вы можете удалить самый "тяжелый" пакет [gnome-control-center](#) (Параметры GNOME 3/41).

По сути, это графическая обертка для gsettings, которая однако достаточно тяжеловесная, и тянет за собой кучу ненужных зависимостей.

```
sudo pacman -Rsn gnome-control-center
```

### 10.2.2 Отключение Tracker 3 в GNOME (НОВЫЙ СПОСОБ)

Tracker - это встроенный поисковик для GNOME, который индексирует все файлы на диске и не только. Как любой индексатор файловых систем, он призван кушать ресурсы и мощности вашего накопителя и висеть в оперативной памяти, хоть и в гораздо меньшей степени чем конкуренты (До Windows, с их 100% загруженности на диск, еще как до луны). Тем не менее, его отключение может положительно повлиять на жизненный цикл вашего HDD (в особенности) или SSD, поэтому его можно отключить в целях профилактики диска. Обратите внимание, что после отключения поиск файлов в GNOME может работать некорректно и не так быстро.

#### Инструкция по отключению

```
systemctl --user mask tracker-miner-apps tracker-miner-fs tracker-store
```

После перезагрузки:

```
rm -rf ~/.cache/tracker ~/.local/share/tracker # Чистим кэш tracker  
tracker daemon -t # Проверяем, должно быть 0 PID
```

### 10.2.3 Отключение ненужных GSD служб GNOME (НОВЫЙ СПОСОБ)

**Внимание:** Способ отключения служб был обновлен. Крайне рекомендуется использовать именно новый способ через systemd взамен старого, опасного переименования библиотек.

GSD (gnome-settings-daemon) службы, это, как следует из названия, службы настройки GNOME и связанных приложений. Если отойти от строго определения, то это просто службы-настройки на все случаи жизни, которые просто висят у вас в оперативной памяти в ожидании когда вам, или другому приложению, к примеру, понадобиться настроить/интегрировать поддержку планшета Wacom или других устройств. И другие подобные вещи.

# Отключение службы интеграции GNOME с графическим планшетом Wacom. Если у вас такого нет - смело отключайте.

```
systemctl --user mask org.gnome.SettingsDaemon.Wacom.service
```

# Отключение службы уведомления печати. Если нет принтера - отключаем.

```
systemctl --user mask org.gnome.SettingsDaemon.PrintNotifications.service
```

# Отключение службы управления цветовыми профилями GNOME. Отключив её не будет работать тёплый режим экрана (Системный аналог Redshift).

```
systemctl --user mask org.gnome.SettingsDaemon.Color.service
```

# Отключение службы управления специальными возможностями системы. **Не отключать людям с ограниченными возможностями!**

```
systemctl --user mask org.gnome.SettingsDaemon.AllySettings.service
```

# Отключает службу управления беспроводными интернет-соединениями. Не рекомендуется отключать для ноутбуков с активным использованием Wi-Fi.

```
systemctl --user mask org.gnome.SettingsDaemon.Wwan.service
```

# Отключение службы защиты от неавторизованных USB устройств при блокировке экрана.

```
systemctl --user mask org.gnome.SettingsDaemon.UsbProtection.service
```

# Отключаем службу настройки автоматической блокировки экрана.

```
systemctl --user mask org.gnome.SettingsDaemon.ScreensaverProxy.service
```

# Отключение службы настройки общественного доступа к файлам и директориям.

```
systemctl --user mask org.gnome.SettingsDaemon.Sharing.service
```

# Отключение службы управления подсистемой rfkill, отвечающей за отключения любого радиопередатчика в системе (сюда же относятся Wi-Fi и Bluetooth, поэтому данная служба нужна, скорее всего, для режима в "самолете").

```
systemctl --user mask org.gnome.SettingsDaemon.Rfkill.service
```

# Отключение службы управления клавиатурой и раскладками GNOME. Можно смело отключать если уже настроили все раскладки и настройки клавиатуры заранее.

```
systemctl --user mask org.gnome.SettingsDaemon.Keyboard.service
```

# Отключаем службу управления звуком GNOME 3/40. Отключает **ТОЛЬКО** настройки звука GNOME 3/40, а не вообще все управлением звуком в системе.

```
systemctl --user mask org.gnome.SettingsDaemon.Sound.service
```

# Отключение службы интеграции GNOME с картридером.

```
systemctl --user mask org.gnome.SettingsDaemon.Smartcard.service
```

# Отключение службы слежения за свободным пространством на диске.

```
systemctl --user mask org.gnome.SettingsDaemon.Housekeeping.service
```

# Отключение службы управления питанием в GNOME. Можете оставить эту службу включенной, в случае если у вас ноутбук.

```
systemctl --user mask org.gnome.SettingsDaemon.Power.service
```

# Отключение служб Evolution для синхронизации онлайн аккаунтов (Если вы конечно не удалили сам Evolution через команду чистки мусора выше)

```
systemctl --user mask evolution-addressbook-factory evolution-calendar-factory  
evolution-source-registry
```

Если после отключения какой-либо из вышеперечисленных служб что-то пошло не так, или просто какую-либо из них понадобилось включить, просто пропишите:

```
systemctl --user unmask --now СЛУЖБА
```

Служба вернется в строй после перезагрузки.

**Внимание:** Если вы по-прежнему использовали старый способ с переименованием библиотек, то настоятельно рекомендуется выполнить переустановку пакета gnome-settings-daemon, а затем выполнить отключение ненужных вам служб уже новым способом.

#### **10.2.4 gnome-shell-performance и mutter-performance**

Пакеты `gnome-shell-performance` и `mutter-performance` - это модифицированные версии пакетов GNOME, где упор сделан на плавность и отзывчивость благодаря включению большого количества патчей для повышения производительности DE.

##### **Установка gnome-shell-performance**

```
git clone https://aur.archlinux.org/gnome-shell-performance.git # Загружаем исходники
cd gnome-shell-performance                                     # Переход в директорию
makepkg -srsc                                                 # Сборка и установка
```

### Установка `mutter-performance`

```
git clone https://aur.archlinux.org/mutter-performance.git # Загружаем исходники
cd mutter-performance                                       # Переход в директорию
makepkg -srsc                                               # Сборка и установка
```

Также можно выполнить компиляцию пакетов при помощи Clang: Mesa (Только для оборудования Intel & AMD), Wayland, Wayland-protocols, Lib32-wayland, Egl-wayland, xorg-server и многих других.

Более подробную информацию вы можете найти в разделе "Общее ускорение системы".

## 10.2.5 Бонус: немного косметики

С обновлением GNOME 42 некоторые приложения на GTK 4 стали использовать тему libadwaita, но из-за этого приложения на GTK 3 стали выглядеть неоднородными, не говоря уж о Qt.

Чтобы это исправить, установите портированную тему libadwaita для GTK 3.

### Установка

```
git clone https://aur.archlinux.org/adw-gtk3.git # Скачиваем исходники
cd adw-gtk3                                      # Переход в директорию
makepkg -srsc                                      # Сборка и установка

# Устанавливаем как тему по умолчанию
gsettings set org.gnome.desktop.interface gtk-theme adw-gtk3
```

## 10.2.6 Результат

По окончании всех оптимизаций мы получаем потребление на уровне современной XFCE, но в отличие от оной уже на современном GTK4, а также со всеми рабочими эффектами и анимациями.

Имя процесса	Процессы		Ресурсы		Файловые системы			Запись диска	Приоритет
	Пользовател	% ЦП	ID	Память	Суммарное чтение	Суммарная запись	Чтение диска		
gnome-shell	vasily	1,45	2491	230,3 МБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
Xorg	vasily	1,45	2439	22,0 МБ	59,0 МБ	413,7 кБ	Н/Д	Н/Д	Очень высоки
gnome-system-monitor	vasily	0,18	3086	17,3 МБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
gnome-screenshot	vasily	2,18	4192	10,6 МБ	127,0 кБ	Н/Д	Н/Д	Н/Д	Обычный
gsd-xsettings	vasily	0,00	2598	7,6 МБ	98,3 кБ	Н/Д	Н/Д	Н/Д	Обычный
gsd-media-keys	vasily	0,00	2594	5,9 МБ	323,6 кБ	Н/Д	Н/Д	Н/Д	Обычный
gjs	vasily	0,00	2579	5,1 МБ	16,4 кБ	Н/Д	Н/Д	Н/Д	Обычный
pulseaudio	vasily	0,00	2541	4,3 МБ	1,9 МБ	41,0 кБ	Н/Д	Н/Д	Очень высоки
at-spi-bus-launcher	vasily	0,00	2458	2,8 МБ	28,7 кБ	Н/Д	Н/Д	Н/Д	Обычный
gnome-session-binary	vasily	0,00	2446	2,5 МБ	38,3 МБ	41,0 кБ	Н/Д	Н/Д	Обычный
gsd-datetime	vasily	0,00	2600	2,2 МБ	53,2 кБ	Н/Д	Н/Д	Н/Д	Обычный
gnome-keyring-daemon	vasily	0,00	2481	1,1 МБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
gsd-housekeeping	vasily	0,00	2599	954,4 кБ	233,5 кБ	Н/Д	Н/Д	Н/Д	Обычный
at-spi2-registryd	vasily	0,00	2586	843,8 кБ	77,8 кБ	Н/Д	Н/Д	Н/Д	Обычный
gsettings-helper	vasily	0,00	2558	831,5 кБ	16,4 кБ	Н/Д	Н/Д	Н/Д	Обычный
dconf-service	vasily	0,00	2699	704,5 кБ	118,8 кБ	32,8 кБ	Н/Д	Н/Д	Обычный
startx	vasily	0,00	2416	598,0 кБ	2,6 МБ	94,2 кБ	Н/Д	Н/Д	Обычный
dbus-daemon	vasily	0,00	2453	553,0 кБ	45,1 кБ	Н/Д	Н/Д	Н/Д	Обычный
dbus-launch	vasily	0,00	2452	303,1 кБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
sh	vasily	0,00	4170	282,6 кБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
dbus-daemon	vasily	0,00	2463	221,2 кБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
sh	vasily	0,00	4168	204,8 кБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
xinit	vasily	0,00	2438	155,6 кБ	24,6 кБ	Н/Д	Н/Д	Н/Д	Обычный
sleep	vasily	0,00	4172	90,1 кБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный

## Видеоверсия

<https://www.youtube.com/watch?v=YIViA-nOzsg>

## Демонстрация плавности

<https://www.youtube.com/watch?v=1TjicRvrFbo>

## 10.3 KDE Plasma 5

Несмотря на то, что авторы ARU считают эту оболочку довольно перегруженной, она по прежнему остается лидером по меньшему энергопотреблению оперативной памяти среди других рабочих окружений. Однако, "бесконечность - не предел", поэтому в этом разделе мы сделаем так, чтобы ваша plasma-shell кушала еще меньше аппаратных ресурсов, и применим на ней другие твики.

### 10.3.1 Удаление мусора из Plasma 5

```
sudo pacman -Rsn kwayland-integration kwallet-pam plasma-thunderbolt plasma-vault
    ↪ powerdevil plasma-sdk kgamma5 drkonqi discover oxygen bluedevil plasma-browser-
    ↪ integration plasma-firewall
# Не удаляйте powerdevil если у вас ноутбук, а bluedevil если используете bluetooth
    ↪ соответственно.

sudo pacman -Rsn plasma-pa      # Удаляем виджет управления звуком.
sudo pacman -S kmix             # Замена виджету plasma-pa, совместим с ALSA.
```

**P.S.** Удаляйте пакеты с осознанием того, что вы делайте. Несмотря на то, что здесь были собраны наиболее сомнительные по соотношению нужности/прожорливости пакеты, вы можете найти какой-либо из пакетов полезным и нужным.

**Предупреждение:** Некоторые пакеты из вышеприведенной команды могут быть не найдены в вашей системе. В таком случае просто выпишите их из команды.

### 10.3.2 Отключение Baloo в Plasma

Baloo - это файловый индексатор в Plasma, аналог Tracker в GNOME, который однако **ОЧЕНЬ прожорливый**, и ест довольно много ресурсов процессора и памяти, вдобавок фоном нагружая ваш диск, в отличии от того же Tracker 3. Поэтому, мы рекомендуем отключать его в любом случае, HDD у вас, или SSD. Хоть разработчики и пытались исправить ситуацию с его непомерным потреблением ресурсов, по прежнему **осталась проблема "утечки" оперативной памяти среди подпроцессов Baloo**.

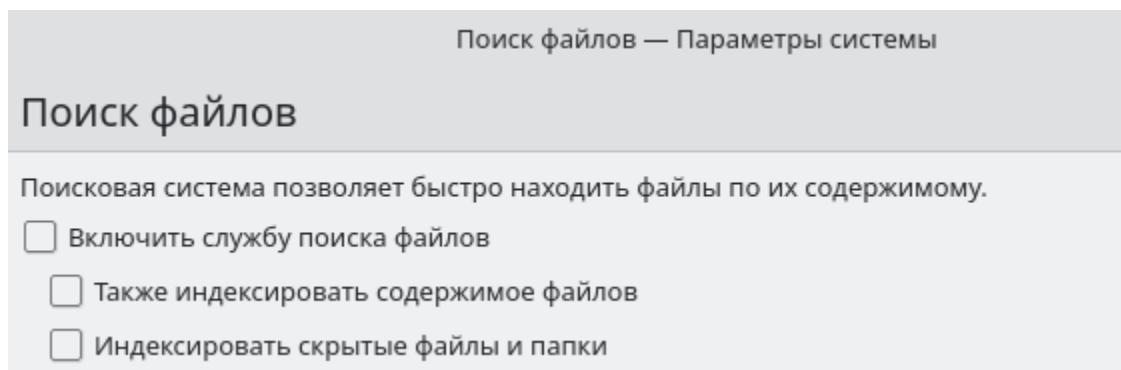
#### Инструкция по отключению:

```
systemctl --user mask kde-baloo.service          # Полное отключение
systemctl --user mask plasma-baloorunner.service
```

Или:

```
balooctl suspend           # Усыпляем работу индексатора
balooctl disable          # Отключаем Baloo
balooctl purge             # Чистим кэш
```

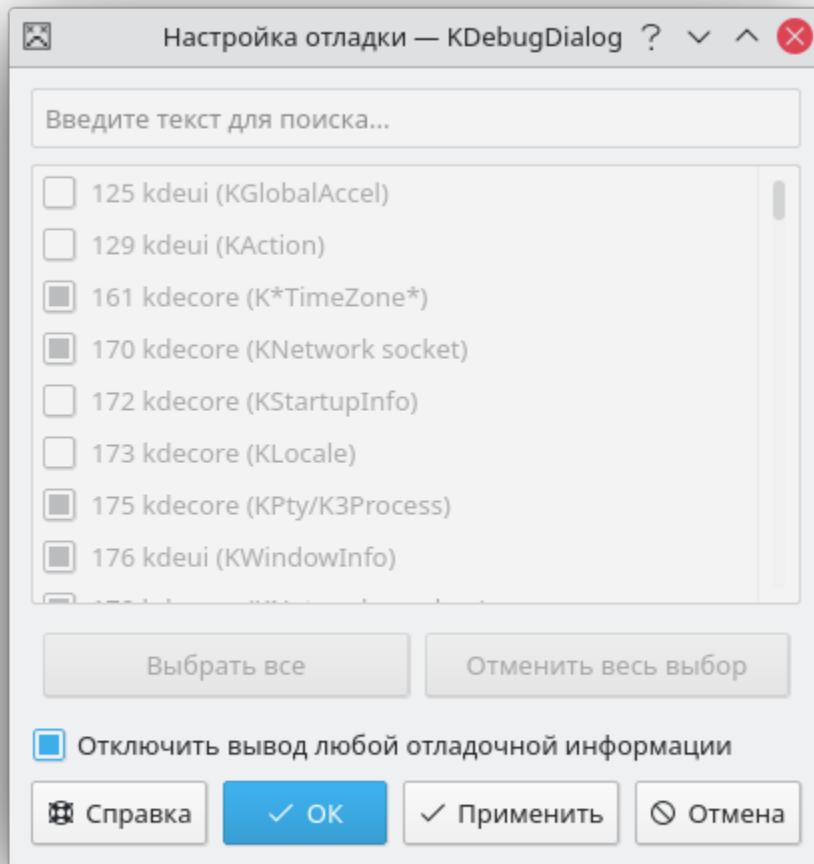
Его так же можно отключить в графических настройках Plasma:



### 10.3.3 Отключение отладочной информации в KDE 5

Слышали о таких настройках отладки в KDE? Нет? Вот и мы не слышали, а они есть. Так как рядовой пользователь почти не видит этой самой "отладочной информации", мы считаем что лучше отключить её вывод и не тратить на это процессорное время. Чтобы это сделать, введите в терминал или меню запуска приложений команду `kdebugdialog5`. Перед вами появится диалоговое окно, где вам нужно поставить галочку на пункте "*Отключить вывод любой отладочной информации*". Затем, просто нажимаете "*Применить*" и "*OK*".

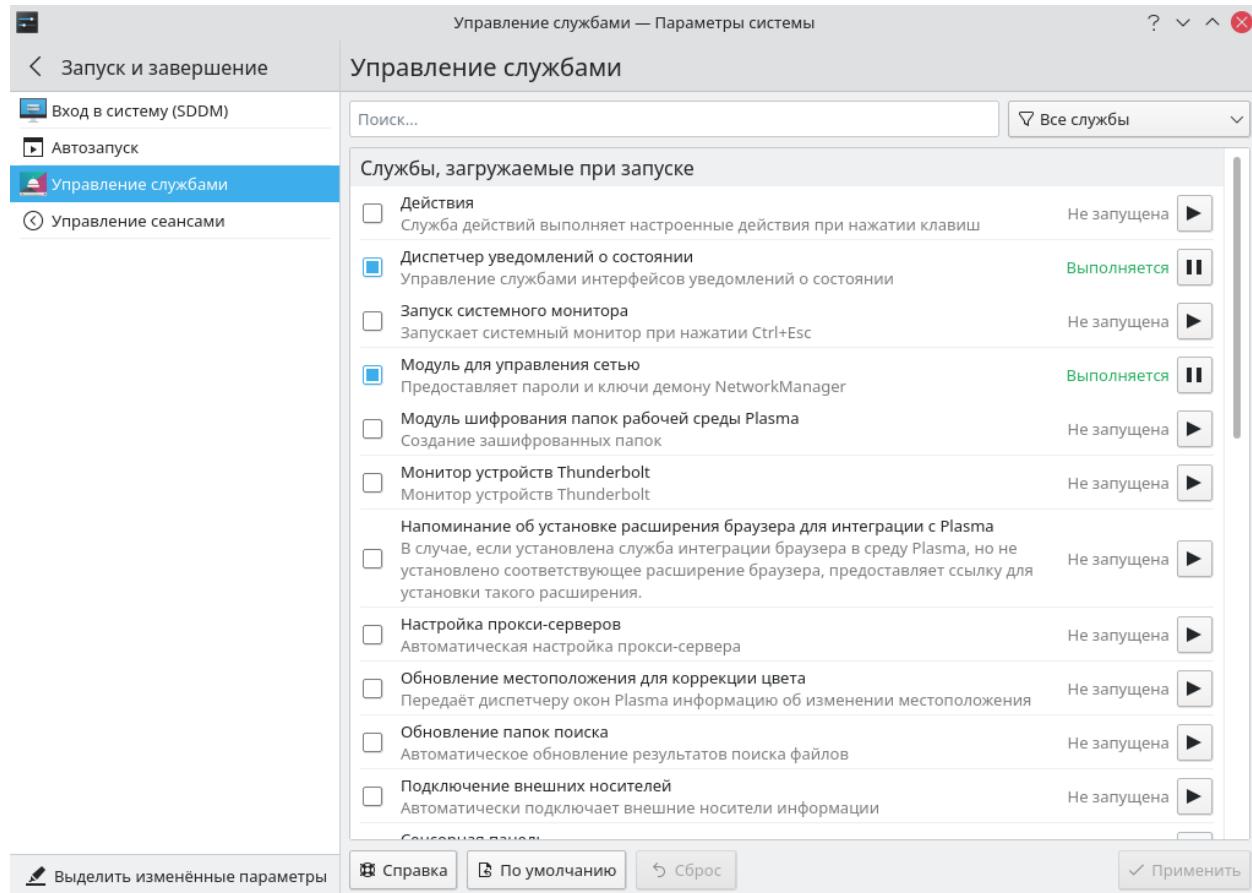
Сбор отладочной информации теперь отключен.



### 10.3.4 Отключение ненужных служб Plasma

По аналогии с GNOME, у Plasma тоже есть свои службы настройки, которые хоть и гораздо менее требовательны к ресурсам. Тем не менее, это по прежнему солянка из различных процессов, которые вам далеко не всегда пригодятся, а отключая ненужные из них вам службы вы можете чуть снизить потребление оперативной памяти вашей оболочкой, т.к. по умолчанию все службы включены.

Настройка служб происходит в графических настройках Plasma, в разделе "Запуск и завершение" -> "Управление службами"



#### Список служб к отключению:

*Монитор устройств Thunderbolt* -> Отключаем, если вы не используете Thunderbolt

*Запуск системного монитора* -> Отключаем, довольно бесполезная служба.

*Напоминание, об установке расширения браузера* -> Еще более бесполезная служба, отключаем.

*Настройка прокси-серверов* -> Отключайте если не используете прокси/системный VPN.

*Bluetooth* -> Отключайте если не используете bluetooth (Если удален bluedevil, этого пункта может и не быть).

*Учётные записи* -> Нужна только если у вас больше одной учетной записи на компьютере.

*Сенсорная панель* -> Отключаем если её нет или вы ей не пользуетесь.

*KScreen 2* -> Нужна только мультимониторным конфигурациям, если у вас один монитор - отключайте.

*Обновление местоположения для коррекции цвета* -> Нужна для "теплого режима" экрана, аналог Redshift. Если не пользуетесь или в ваш монитор встроен этот режим - отключайте.

*Модуль шифрования папок рабочей среды Plasma* -> Нужна только если вы параноик. Впрочем, параноики используют более тяжёлые средства шифрования, поэтому отключаем.

*Слежение за изменениями в URL* -> Работает только в сетевых папках, если вы ими не часто пользуетесь - отключаем.

*Слежение за свободным местом на диске* -> Вещь полезная, но это вы можете сделать и самостоятельно через виджеты, поэтому Откл./Оставлять по желанию.

*SMART* -> Тоже довольно полезная служба, поэтому отключайте на свое усмотрение.

*Диспетчер уведомлений о состоянии* -> Нужна для правильной работы лотка и трея.

*Служба синхронизации параметров GNOME/GTK* -> Осуществляет смену GTK темы на лету. Если отключите, смена GTK темы будет применяться только после перезагрузки.

*Фоновая служба клавиатуры* -> Служба для отображения раскладки в системном лотке.

*Служба локальных сообщений* -> Следит в общении между терминалами через команды wall и write. Это очень специфично, поэтому отключаем.

*Модуль для управления сетью* -> Добавляет системный лоток виджет для управления сетевыми подключениями. Отключайте, если не используете NetworkManager.

*Состояние сети* -> Оповещает приложения в случае неработоспособности интернет-соединения. Тоже довольно нишевая служба, можно отключить.

*Подключение внешних носителей* -> Автоматически примонтирует внешние устройства при их подключении. Например, такие как USB-флешки. Отключайте на свое усмотрение.

*Часовой пояс* -> Информирует другие приложения об изменении системного часового пояса. Довольно редко применимо, можно отключить.

*Обновление папок поиска* -> Автоматически обновляет результат поиска файлов. Отключаем на свое усмотрение. Кроме того, судя по всему работает только в Dolphin.

*Действия* -> Обеспечивает работу специально назначенных действий в настройках. Если вы не используйте кастомные бинды, можете отключить.

*Фоновая служба меню приложений* -> Странная служба. По своей функции она осуществляет обновление Меню Приложений при появлении новых ярлыков, однако даже при её отключении этот функционал работает. Отключайте на свое усмотрение.

### 10.3.5 Настройка работы KWin для увеличения плавности

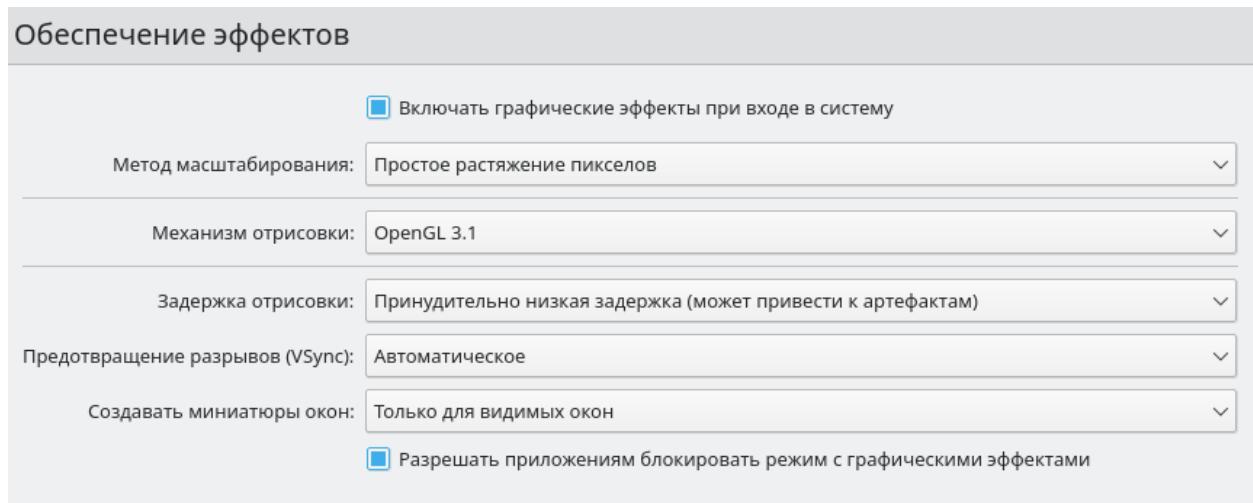
До недавнего времени у Plasma были определенные проблемы с качеством отрисовки и работой композитора в целом. Были и серьёзные проблемы при работе с закрытым драйвером NVIDIA. Правда, начиная с версии плазмы 5.21, ситуация значительно улучшилась, но по прежнему довольно нестабильна. Напомним, что композитор, и одновременно оконный менеджер, в Plasma это kwin - и он отвечает за:

1. Управление окнами, и все что с ними связано.
2. Различные графические эффекты и визуальные "приблуды" (Прозрачность, тени, размытие и проч.)
3. Плавность отрисовки и бесшовность отображаемой картинки, т. е. обеспечивает синхронизацию между кадрами (Vsync), предотвращает тириング (разрыв между кадрами).

Вообщем, делает довольно много интересных вещей.

Но нас интересует только третья и немного вторая его функции.

Итак, чтобы обеспечить наилучшую плавность и визуальное качество отклика, нам нужно провести грамотную его (композитора) настройку. Для этого мы перейдем в соответствующий раздел настроек Plasma, т. е. в Экран -> Обеспечение Эффектов.



Что-ж, давайте по порядку.

#### "Включать графические эффекты при входе в систему"

Данная опция отвечает за то, будет ли композитор брать на себя роль за отрисовку графических эффектов, и синхронизации кадров соответственно. Т. е. будет ли он выполнять свои две последние функции (См. выше) сразу после запуска оболочки. Вы можете отключить этот параметр, в случае крайней экономии аппаратных ресурсов, т.к. это снижает с композитора роль за граф. эффекты и вертикальную синхронизацию, то это также может уменьшить его потребление ресурсов компьютера вдвое, и он просто станет лишь менеджером управления окнами в Plasma.

#### "Механизм отрисовки"

Отвечает за то, средствами какого API-бэкенда будет производиться отрисовка. OpenGL механизм дает больше возможностей для обеспечения различных графических эффектов, и лучшую синхронизацию кадров. Принципиальной разницы между OpenGL 2.0 и

OpenGL 3.1 - нет. Поддержка OpenGL 2.0 нужна и остается только для работы со старыми видеокартами, у которых нет поддержки OpenGL 3.1. XRender механизм довольно старомоден, и является серьёзно морально устаревшим, он не поддерживает такое же количество граф. эффектов как OpenGL, поэтому не удивляйтесь что какие-то из них не будут работать на этом механизме отрисовки. Кроме того, с этим бэкендом не работает синхронизация кадров, т. е. Vsync автоматически отключается при выборе данного механизма, и может появиться тириング. Тем не менее, XRender обеспечивает практически минимальное потребление оперативной памяти компьютера со стороны композитора, и полагается в основном на ресурсы центрального процессора, практически не действуя видеокарту и не создавая задержки ввода. Поэтому он может эффективно использоватьсь в комбинации с включенной "Tearfree" опцией открытого драйвера AMD/Intel исправляющей тириинг, и "ForceCompositionPipeline" закрытого драйвера NVIDIA (Что, впрочем, не очень рекомендуется при наличии OpenGL бэкенда с поддержкой Vsync) или NVIDIA PRIME Sync (В таком случае даже рекомендуется его использовать, т.к. это может исправить проблему высокой задержки на ноутбуках с поддержкой NVIDIA PRIME, а проблема тириинга при этом будет решаться использованием самой технологии PRIME Sync). И конечно для AMD Freesync и Nvidia Gsync.

#### **"Задержка отрисовки"**

Параметр напрямую влияющий на плавность отрисовки и синхронизацию между кадрами. Он задает с какой задержкой композитор перейдет к композитингу и синхронизации следующего кадра. Соответственно, чем меньше задержка между этими событиями, тем быстрее композитор сможет нарисовать последующие кадры, благодаря чему и достигается такое расплывчатое понятие, как "плавность" картинки, отсутствие высокой задержки ввода (input lag) и в тоже время бесшовность картинки, т.е. отсутствие тиринга. Лучшим вариантом для закрытого драйвера NVIDIA будет, и настоятельно рекомендуется - "Принудительно низкая задержка". Для открытых драйверов Intel/AMD не все так однозначно, и с принудительно низкой задержкой могут возникать артефакты отрисовки. Тем не менее, все также рекомендуется "Предпочитать низкую задержку".

#### **"Предотвращение разрывов (VSync)"**

Здесь, мы выбираем метод с которым будут синхронизироваться наши кадры (VSync). Лучше всего отдать его предпочтение автоматическому выбору самого композитора под ваш видеодрайвер, т. е. "Автоматически". Можно также отдать предпочтение методу "При минимуме затрат", где следуя из названия, будут достигаться минимальные затраты на синхронизацию кадра. Однако, этот метод работает только при обновлении всего экрана, например при воспроизведении видео. Поэтому при его использовании может "проявляться" тириинг в некоторых местах при частичном обновлении экрана. Другие методы могут ухудшать производительность, либо в целом, либо для определенных видеодрайверов ("Повторное использование" ухудшает производительность при использовании с драйверами Mesa, т.е. на оборудовании с Intel/AMD).

#### **"Разрешить приложениям блокировать режим с графическими эффектами"**

Не всегда, и не во всех приложениях нужно осуществлять композитинг и отрисовку графических эффектов, поэтому была сделана эта опция чтобы дать разрешение на их блокировку другими приложениями. В целом, блокировка графических эффектов нужна в основном для полноэкранных видеоигр, чтобы не создавать для них лишней задержки ввода и немного улучшить их производительность. Настоятельно рекомендуется оставлять включенным данный параметр.

#### **"Метод масштабирования"**

Из названия понятно, что это метод с которым у вас будет масштабироваться интерфейс.

"Простое растяжение пикселов" - Самый производительный метод, но в тоже время

самый топорный по качеству.

"Со сглаживанием" - оптимальный вариант, и рекомендуется большинству конфигураций.

"Точное сглаживание" - Лучший вариант с точки зрения качества, но при этом жертвует некоторой производительностью, и этот метод может работать не со всеми видеокартами и приводить к артефактам отрисовки.

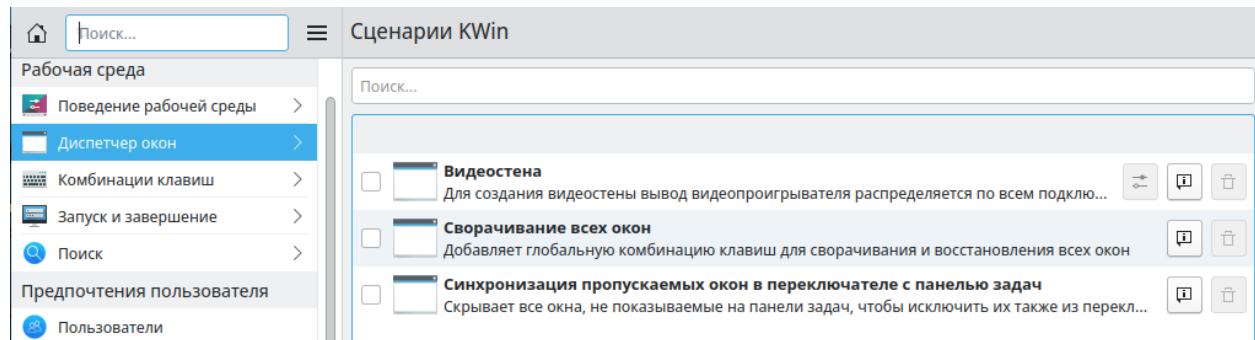
### 10.3.6 Отключение композитинга для полноэкраных окон

`kwin-autocomposer` - расширение для Kwin, которое позволяет полностью отключить композитинг для полноэкраных окон в X11 сессии Plasma. Это помогает исправить дрожание фреймтайма во время игры и понизить задержки.

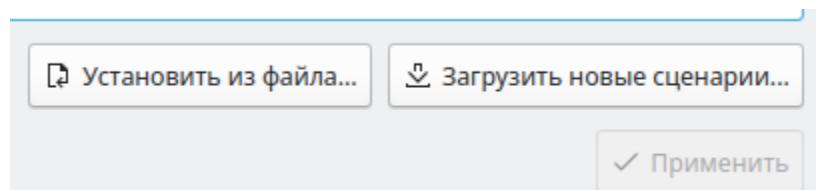
Для Wayland сессий Plasma с версии 5.22 отключение композитинга полноэкраных окон происходит по умолчанию.

#### Установка

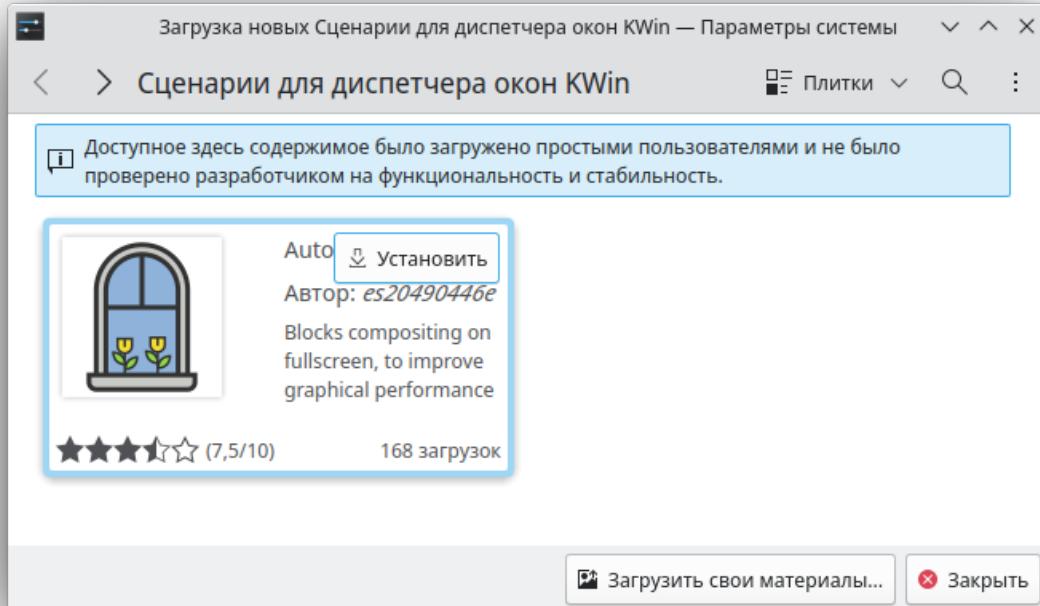
Зайдите в настройки, затем в раздел *Диспетчер окон* -> *Сценарии Kwin*.



Внизу найдите кнопку "Загрузить новые сценарии"



Найдите в представленном каталоге "Autocomposer" выполните его установку.

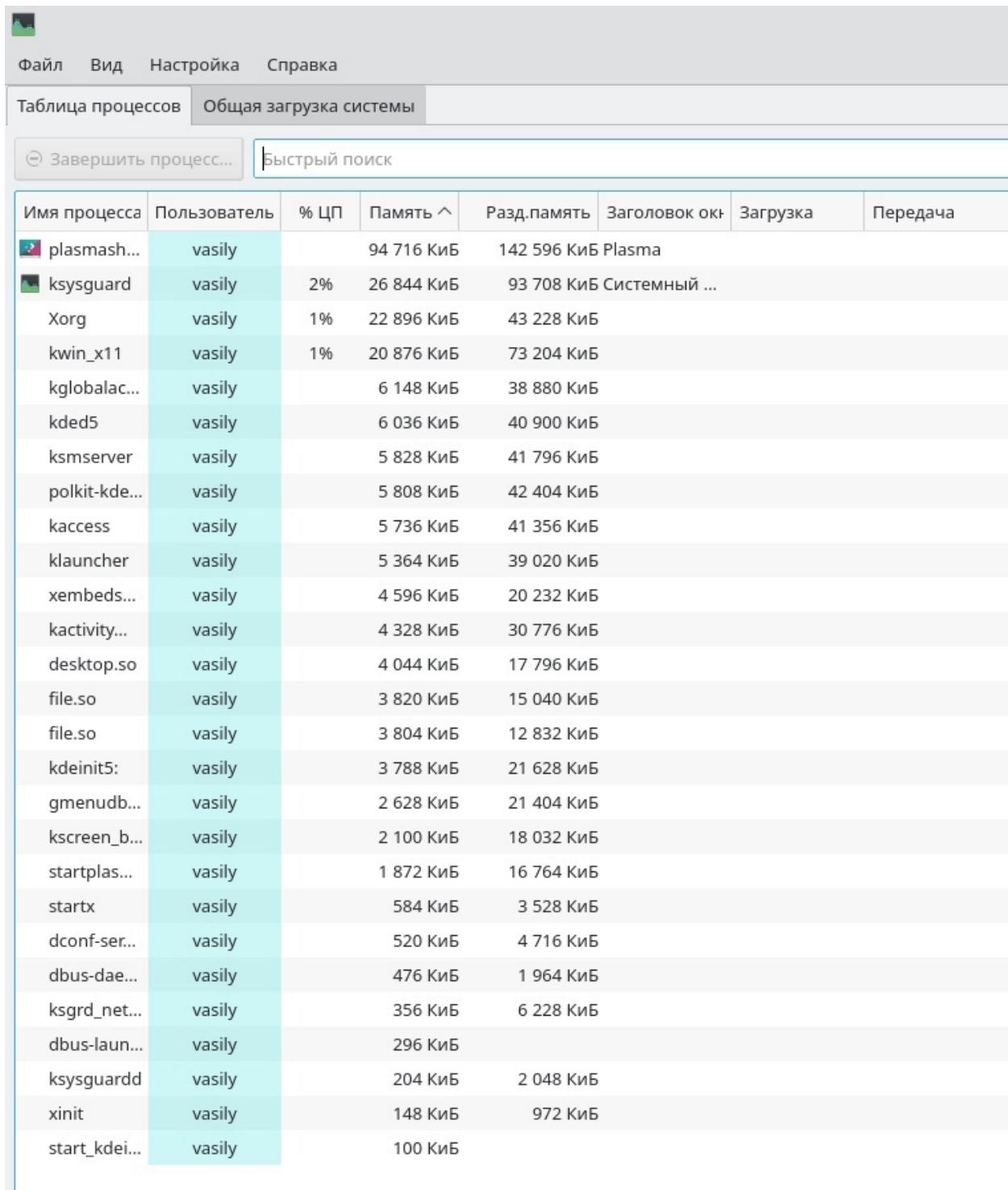


После этого перезагрузите рабочее окружение. Готово.

### 10.3.7 Отключение ненужных графических эффектов Plasma

Plasma предоставляет возможность использовать много различных графических эффектов (С включенным методом отрисовки OpenGL естественно). Но далеко не все из них нужны, и, по сути, являются сугубо декоративным элементом, которые при этом потребляют некоторые мощности оперативной памяти и GPU на их отрисовку. Поэтому, если вы хотите минимизировать потребление этих ресурсов, рекомендуется либо полностью, либо частично отключить графические эффекты. Осуществить это можно, либо как уже говорилось выше, сняв галочку с "Включать графические эффекты при входе в систему" в настройках Plasma "Экран -> Обеспечение эффектов", либо можно частично отключить определенные граф. эффекты в настройках "Поведение рабочей среды" -> "Эффекты". Какие из них оставлять, а какие нет - решать только вам, но чем меньше эффектов будет включено, тем меньше потребление ресурсов.

### 10.3.8 Результат



The screenshot shows the KDE Task Manager window. At the top, there is a menu bar with 'Файл', 'Вид', 'Настройка', and 'Справка'. Below the menu is a tab bar with 'Таблица процессов' (selected) and 'Общая загрузка системы'. A search bar at the top right contains the placeholder 'Быстрый поиск'. The main area is a table listing processes:

Имя процесса	Пользователь	% ЦП	Память ^	Разд.память	Заголовок окн	Загрузка	Передача
plasmash...	vasily		94 716 КиБ	142 596 КиБ	Plasma		
ksysguard	vasily	2%	26 844 КиБ	93 708 КиБ	Системный ...		
Xorg	vasily	1%	22 896 КиБ	43 228 КиБ			
kwin_x11	vasily	1%	20 876 КиБ	73 204 КиБ			
kglobalac...	vasily		6 148 КиБ	38 880 КиБ			
kded5	vasily		6 036 КиБ	40 900 КиБ			
ksmserver	vasily		5 828 КиБ	41 796 КиБ			
polkit-kde...	vasily		5 808 КиБ	42 404 КиБ			
kaccess	vasily		5 736 КиБ	41 356 КиБ			
klauncher	vasily		5 364 КиБ	39 020 КиБ			
xembeds...	vasily		4 596 КиБ	20 232 КиБ			
kactivity...	vasily		4 328 КиБ	30 776 КиБ			
desktop.so	vasily		4 044 КиБ	17 796 КиБ			
file.so	vasily		3 820 КиБ	15 040 КиБ			
file.so	vasily		3 804 КиБ	12 832 КиБ			
kdeinit5:	vasily		3 788 КиБ	21 628 КиБ			
gmenudb...	vasily		2 628 КиБ	21 404 КиБ			
kscreen_b...	vasily		2 100 КиБ	18 032 КиБ			
startplas...	vasily		1 872 КиБ	16 764 КиБ			
startx	vasily		584 КиБ	3 528 КиБ			
dconf-ser...	vasily		520 КиБ	4 716 КиБ			
dbus-dae...	vasily		476 КиБ	1 964 КиБ			
ksgrd_net...	vasily		356 КиБ	6 228 КиБ			
dbus-laun...	vasily		296 КиБ				
ksysguardd	vasily		204 КиБ	2 048 КиБ			
xinit	vasily		148 КиБ	972 КиБ			
start_kdei...	vasily		100 КиБ				

## 10.4 Xfce4

Xfce, или мышонок в простонародье, является примером "старой школы" среди всех рабочих окружений. Он до сих пор сохранил свою незамысловатость и простоту, однако с последними выпусками и переходом на GTK3 к сожалению потерял свою легковесность. Поэтому в этом разделе, мы поговорим об оптимизации Xfce.

### 10.4.1 Удаление потенциально ненужных компонентов Xfce

Честно говоря, в Xfce довольно мало откровенно "ненужных" пакетов. И, по сути, все сводиться к личным предпочтениям, какие пакеты вам нужны, а какие нет. Поэтому рассматриваете указанные ниже инструкции по удалению на свой лад.

```
# Удалит менеджер питания Xfce. Нужен только если у вас ноутбук и нужно настроить энергосбережение. На ПК можно считать это лишним фоновым процессом который висит у вас в памяти.
sudo pacman -Rn xfce4-power-manager

# Пожалуй единственный, действительно мусорный пакет, который весит процессом на случай если вам нужно будет "найти приложение", которые вы можете и сами найти в соответствующем меню.
sudo pacman -Rsn xfce4-appfinder

# Набор тем для Xfwm (Оконного менеджера по умолчанию в Xfce). Удаляйте по желанию.
sudo pacman -Rsn xfwm4-themes

# Дополнение к Thunar, и фоновый процесс для удобного и скорого управления различными съемными устройствами при их подключении, например такими как USB-флешки, CD диски, камера и пр.. Если такими устройствами не пользуетесь, или делаете это не часто - можете удалять.
sudo pacman -Rsn thunar-volman

# Создает превью изображений различных форматов для Thunar. Довольно прожорливая штука, поэтому если хотите можете его удалить.
sudo pacman -Rsn tumbler

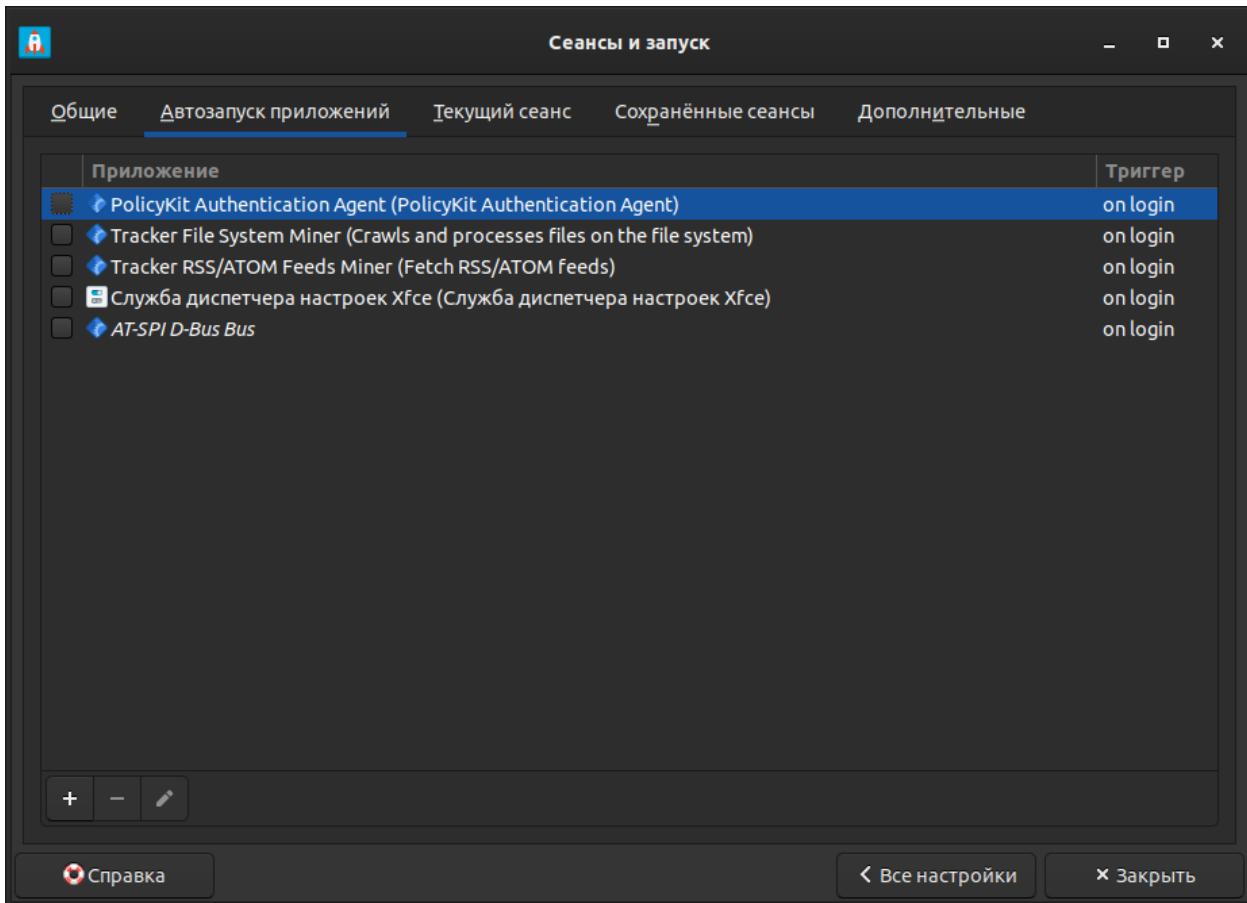
# Терминал по умолчанию для Xfce. Является довольно прожорливым, поэтому можете заменить его на менее энергозатратные аналоги.
sudo pacman -Rsn xfce4-terminal

# Графическая обертка для главной панели настроек Xfce. По желанию можете удалить, и использовать вместо неё xfconf-query.

# Демон отображения уведомлений в Xfce. Можете удалить и заменить на более легковесные аналоги (например, dunst), не забудьте при этом добавить замену в автозагрузку.
sudo pacman -Rsn xfce4-notifyd
```

## 10.4.2 Отключение ненужных служб и приложений автозапуска

В Xfce также не так много различных фоновых служб, скорее их очень мало. Тем не менее, они есть, и не все они лично вам могут быть нужны. Настроить их вы можете в настройках "Сеансы и запуск" -> "Автозапуск приложений". Отключить вы можете почти все, они не очень важны для работоспособности оболочки. Единственное, что вы можете оставить - это "PolicyKit Authentication Agent", для приложений требующих пароль на выполнение действий из под sudo/root. Служба "Tracker File System Miner" - это встроенный файловый индексатор Xfce, его можете либо включить для корректной работы поиска в оболочке и Thunar, либо отключить в целях экономии ресурсов компьютера.

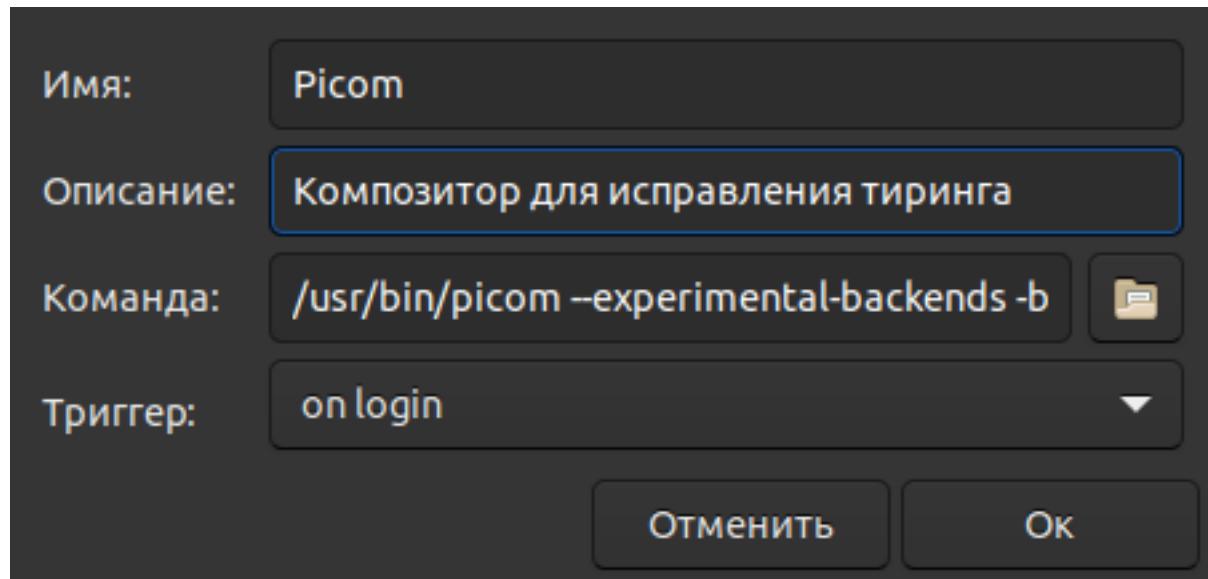


## 10.4.3 Настройка композитора Xfwm4

Композитор по умолчанию в Xfce это Xfwm. К сожалению, порой он достаточно неэффективно выполняет функцию синхронизации кадров (Vsync), поэтому нужно выполнить самостоятельную настройку его работы для исправления проблем тиринга. Сделать это можно в "Редакторе Настроек" -> "xfwm4". Здесь нас интересуют три опции, а именно: "vblank\_mode", "unredirect\_overlays" и "use\_compositing". Теперь подробнее.

```
xfconf-query -c xfwm4 -p /general/unredirect_overlays -s true # Параметр на отвязку полноэкраных окон от работы композитора. В разделе с Plasma эта тема освещалась более подробно. В основном, это применимо к полноэкранным видеоиграм, чтобы не создавать для них лишнюю задержку ввода и немного улучшить их производительность.
```

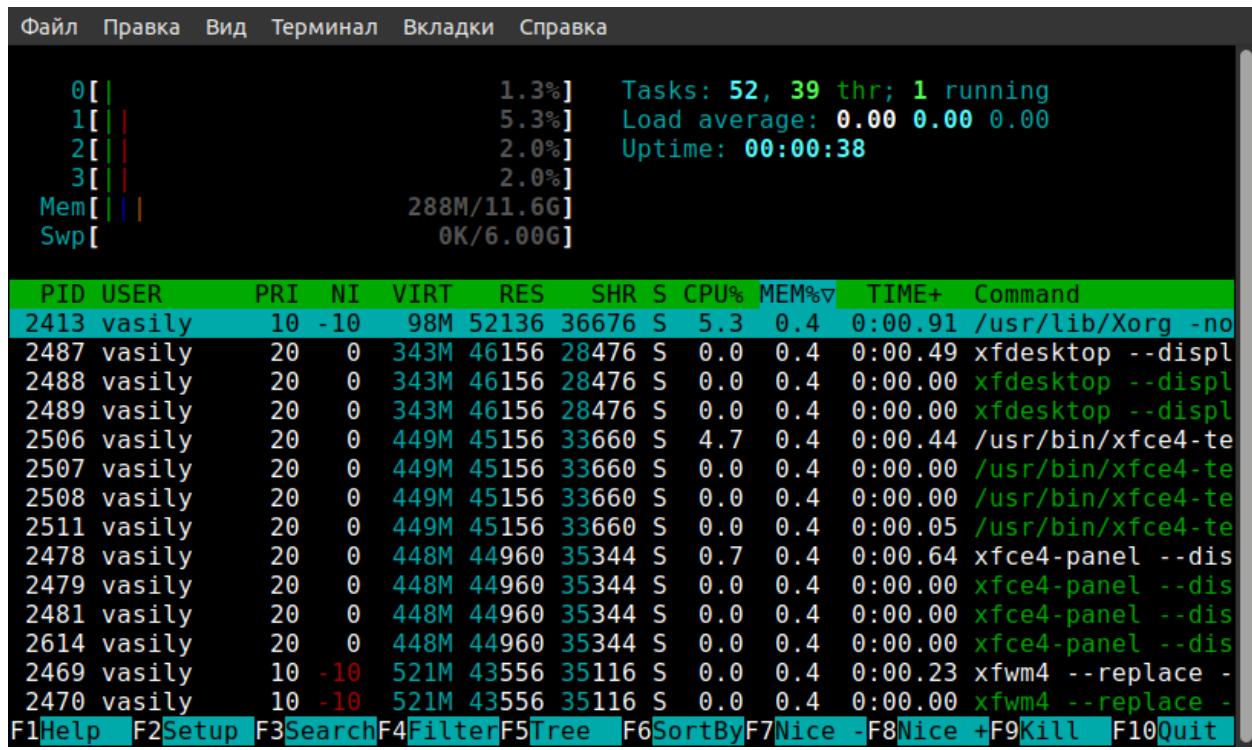
`xfce-query -c xfwm4 -p /general/use_compositing -s true` # Параметр для переключения работы графических эффектов и вертикальной синхронизации композитора. Если отключите (*false*), то Xfwm больше не будет выполнять ни вертикальную синхронизацию, ни отрисовку граф. эффектов, и станет просто оконным менеджером. В целях уменьшения потребления ресурсов, это рекомендуется выключить, однако может снова возникнуть проблема тиринга. Как её решить без применения вертикальной синхронизации было указано ниже, но вы также можете использовать сторонний композитор для решения этой проблемы, например такой как Picom. Чтобы это сделать нужно отключить графические эффекты Xfwm, т.е. как раз выключить параметр *use\_compositing*, и установить `picom` (*sudo pacman -S picom*). И затем добавить его в автозагрузку (См. приложение). Вот и все.



`vblank_mode` задает через какие средства будет осуществляться вертикальная синхронизация кадров. Всего есть три возможных значения:

1. `xfconf-query -c xfwm4 -p /general/vblank_mode -s glx` # Композитинг и синхронизация кадров при помощи OpenGL. Самый надежный вариант для исправления проблем тиринга, как для открытых драйверов, так и (в особенности) для закрытого драйвера NVIDIA. Может создавать некоторую задержку ввода.
2. `xfconf-query -c xfwm4 -p /general/vblank_mode -s xpresent` # Морально устаревший бэкенд отрисовки, который почти не использует ресурсы видеокарты, и перекладывает основную нагрузку за отрисовку эффектов и синхронизации кадров на процессор. В целом, потребление ресурсов с ним меньше чем под glx, и он не создает лишней задержки ввода. И все же, он довольно плохо решает проблему тиринга, поэтому порой он может проявляться. С Закрытым драйвером NVIDIA вертикальная синхронизация при xpresent вообще не будет работать.
3. `xfconf-query -c xfwm4 -p /general/vblank_mode -s off` # Отключение вертикальной синхронизации кадров. Этот вариант можно рассмотреть, в случае если вы компенсируете проблему тиринга через опции драйвера "Tearfree" для Intel/AMD, и "ForceCompistionPipeline" для закрытого драйвера NVIDIA или NVIDIA PRIME Sync (Что даже рекомендуется, т.к. NVIDIA PRIME Sync это единственный возможный способ полного исправления проблемы тиринга на ноутбуках с NVIDIA PRIME, и никакая дополнительная синхронизация обычно не нужна). Также эта опция настоятельно рекомендуется пользователям технологий AMD Freesync и Nvidia G-Sync.

## 10.4.4 Результат



## 10.5 Cinnamon

Cinnamon, или дословно корица, это форк GNOME 3, который был создан разработчиками Linux Mint для исправления проблем своего родителя, когда последний был в крайне нестабильном состоянии. И отчасти им это удалось, но одну из главных проблем GNOME она (корица), к сожалению, унаследовала - это большое потребление оперативной памяти и других ресурсов компьютера. Поэтому здесь мы поговорим об оптимизации нашей булочки с корицей.

### 10.5.1 Отключение ненужных CSD служб (НОВЫЙ СПОСОБ)

Будучи форком GNOME 3, Cinnamon также имеет свой аналог GSD служб, которые называются CSD службами (Cinnamon Settings Daemon). Принципиальных различий от GSD служб у них по сути нет, просто другое название и немного измененный состав.

```
cd ~/.config/autostart # Переходим в директорию автозагрузки
cp -v /etc/xdg/autostart/cinnamon-settings-daemon-*.desktop ./ # Копируем
→автозагрузку службы
```

```
# Отключение службы интеграции Cinnamon с графическим планшетом Wacom. Если у вас его нет - смело отключайте.
```

```
echo "Hidden=true" >> cinnamon-settings-daemon-wacom.desktop
```

```
# Отключение службы интеграции принтера в Cinnamon.
```

```
echo "Hidden=true" >> cinnamon-settings-daemon-print-notifications.desktop
```

# Отключение службы настройки цветовых профилей в Cinnamon.:

```
echo "Hidden=true" >> cinnamon-settings-daemon-color.desktop
```

# Отключение службы настройки "Специальных Возможностей" в Cinnamon. **Не отключать людям с ограниченными возможностями!**

```
echo "Hidden=true" >> cinnamon-settings-daemon-ally-settings.desktop  
echo "Hidden=true" >> cinnamon-settings-daemon-ally-keyboard.desktop
```

# Отключение службы настройки автоматической блокировки экрана.

```
echo "Hidden=true" >> cinnamon-settings-daemon-screensaver-proxy.desktop
```

# Отключаем службу управления звуком Cinnamon. Отключает **ТОЛЬКО** настройки звука Cinnamon, а не вообще все управление звуком в системе.

```
echo "Hidden=true" >> cinnamon-settings-daemon-sound.desktop
```

# Отключение службы интеграции Cinnamon с картридером.

```
echo "Hidden=true" >> cinnamon-settings-daemon-smartcard.desktop
```

# Отключение службы настройки клавиатуры и раскладок Cinnamon. Можно смело выключать если вы уже настроили все раскладки и настройки клавиатуры.

```
echo "Hidden=true" >> cinnamon-settings-daemon-keyboard.desktop
```

# Выключаем службу настройки мониторов Cinnamon. Смело отключайте если у вас нет более одного монитора (ноутбук) и вы настроили герцовку уже имеющихся мониторов.

```
echo "Hidden=true" >> cinnamon-settings-daemon-xrandr.desktop
```

# Отключаем службу автоматического мониторинга внешних, подключаемых устройств. Например таких как USB-флешки, CD диски и прочие внешние носители.

```
echo "Hidden=true" >> cinnamon-settings-daemon-automount.desktop
```

# Отключаем службу слежения за свободным пространством на диске.

```
echo "Hidden=true" >> cinnamon-settings-daemon-housekeeping.desktop
```

# Отключаем службу настройки ориентацией дисплея. Если у вас нет сенсорного экрана или поддержки переворота дисплея - отключайте.:

```
echo "Hidden=true" >> cinnamon-settings-daemon-orientation.desktop
```

# Отключение службы настройки мыши и тачпада Cinnamon.

```
echo "Hidden=true" >> cinnamon-settings-daemon-mouse.desktop
```

# Отключение службы настройки энергосбережения Cinnamon. Можете оставить эту службу если у вас НЕ ноутбук.:

```
echo "Hidden=true" >> cinnamon-settings-daemon-power.desktop
```

# Отключаем службу интеграции работы буфера обмена с Cinnamon.

```
echo "Hidden=true" >> cinnamon-settings-daemon-clipboard.desktop
```

Если после отключения какой-либо из вышеперечисленных служб что-то пошло не так, или просто какую-либо из них понадобилось снова включить, просто пропишите::

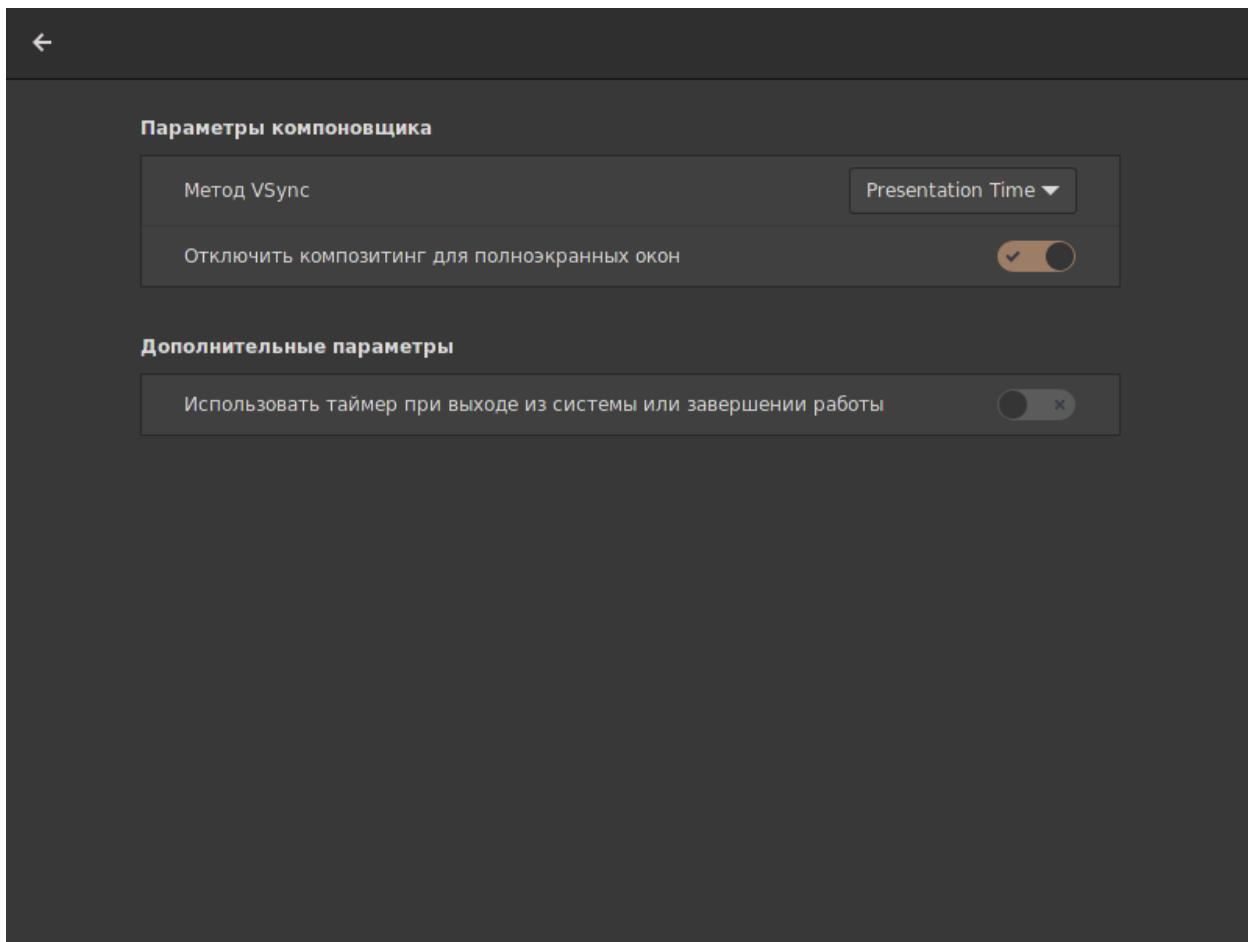
```
rm -rf ~/.config/autostart/cinnamon-settings-daemon-СЛУЖБА.desktop
```

Это вернет нужную службу в строй после перезагрузки.

**Внимание:** Если вы по-прежнему использовали старый способ с переименованием библиотек, то настоятельно рекомендуется выполнить переустановку пакета cinnamon-settings-daemon, а затем выполнить отключение ненужных вам служб уже новым способом.

### 10.5.2 Настройка композитора Muffin

По традиции, настроим композитор оболочки. В случае с Cinnamon это Muffin. Он не содержит много настроек, и его нельзя заменить на другой композитор как мы это делали с Xfwm. По сути, вся настройка Muffin сводится к двум банальным, и уже нам знакомым, параметрам: "*Метод Vsync (Вертикальная Синхронизация)*" и "*Отключение композитора для полноэкраных окон*".



"*Отключение композитора для полноэкранных окон*" - Это уже знакомая вам опция, где из названия все понятно. Вкратце, нужна для уменьшения задержек в видеоиграх создаваемых композитором.

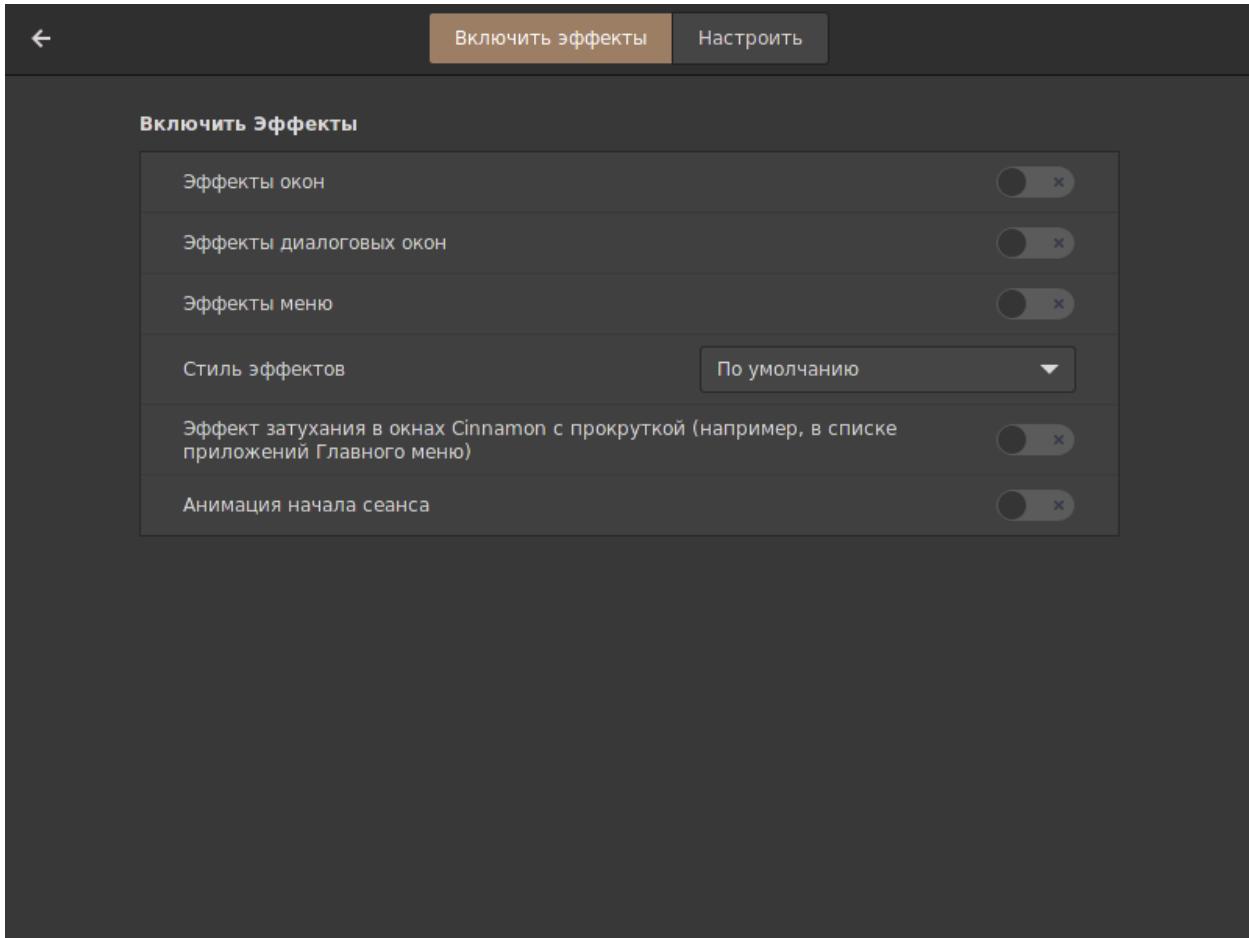
"*Метод Vsync*" - параметр задающий метод синхронизации кадров.

Впрочем, в случае с Muffin, скорее не метод, а ее поведение. Всего есть четыре возможных значения:

1. "None" - Отключение вертикальной синхронизации. Более подробно мы рассматривали применимость этого значения в разделе с Plasma и Xfce. Наиболее рекомендуется пользователям ноутбуков с активированным NVIDIA PRIME Sync или обладателям AMD Freesync и NVIDIA G Sync. Помогает избегать высоких задержек и input lag'a.
2. "*Fallback / Classic*" - Классический метод вертикальной синхронизации, используемый в ранних версиях Cinnamon.
3. "*Swap Throttling*" - Обеспечивает вертикальную синхронизацию с учетом родной частоты обновления вашего монитора. Лучше всего совместим с не-дисплеями (т.е. мониторами).
4. "*Presentation Time*" - Может осуществлять вертикальную синхронизацию сразу нескольких устройств с разной частотой обновления (Герцовой). Рекомендуется включить, если вы используете более одного монитора или дисплея.

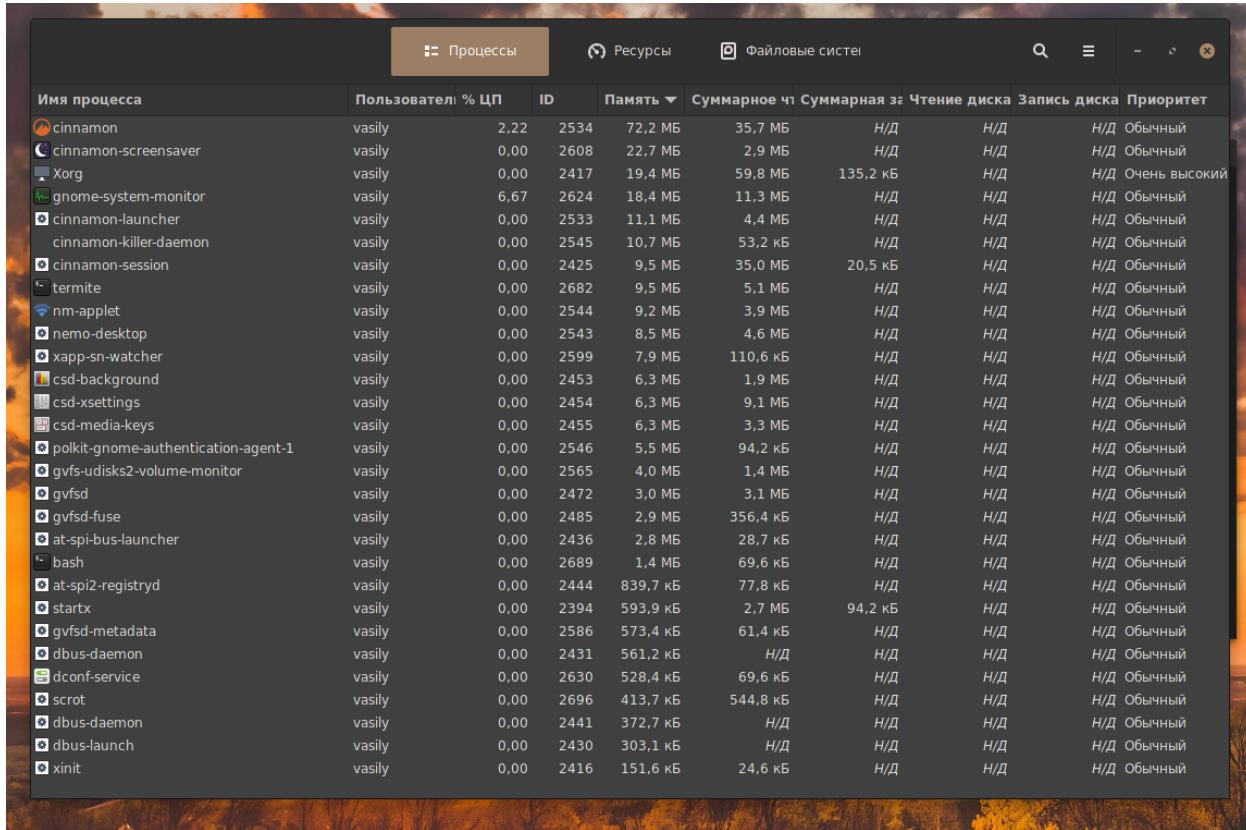
### 10.5.3 Отключение ненужных эффектов Muffin

К сожалению, по умолчанию в Muffin отсутствует опция отключения сразу всех графических эффектов в оболочке (т.е. композитинга). Поэтому, нам нужно отключить их по-очередно в соответствующем разделе настроек "Эффекты".



Желательно, в целях максимальной экономии аппаратных ресурсов, отключить все имеющийся здесь эффекты. Но вы можете сделать это также и выборочно. И как обычно: Чем меньше эффектов включено -> Тем меньше потребление ресурсов ОЗУ и VRAM.

## 10.5.4 Результат



The screenshot shows the Arch Top application window. The title bar has tabs for "Процессы" (Processes), "Ресурсы" (Resources), and "Файловые системы" (Filesystems). The "Процессы" tab is selected. The main area displays a table of processes with the following columns: Имя процесса (Process Name), Пользователь (User), % ЦП (CPU %), ID, Память (Memory), Суммарное чтение (Total Read), Суммарная запись (Total Write), Чтение диска (Disk Read), Запись диска (Disk Write), and Приоритет (Priority). The table lists numerous processes, primarily from the Cinnamon desktop environment, along with system daemons like Xorg, gvfsd, and dbus-daemon. The "vasily" user is the most active, with cinnamonmon at 2.22% CPU usage.

Имя процесса	Пользователь	% ЦП	ID	Память	Суммарное чтение	Суммарная запись	Чтение диска	Запись диска	Приоритет
cinnamonmon	vasily	2,22	2534	72,2 МБ	35,7 МБ	Н/Д	Н/Д	Н/Д	Обычный
cinnamon-screensaver	vasily	0,00	2608	22,7 МБ	2,9 МБ	Н/Д	Н/Д	Н/Д	Обычный
Xorg	vasily	0,00	2417	19,4 МБ	59,8 МБ	135,2 кБ	Н/Д	Н/Д	Очень высокий
gnome-system-monitor	vasily	6,67	2624	18,4 МБ	11,3 МБ	Н/Д	Н/Д	Н/Д	Обычный
cinnamon-launcher	vasily	0,00	2533	11,1 МБ	4,4 МБ	Н/Д	Н/Д	Н/Д	Обычный
cinnamon-killer-daemon	vasily	0,00	2545	10,7 МБ	53,2 кБ	Н/Д	Н/Д	Н/Д	Обычный
cinnamon-session	vasily	0,00	2425	9,5 МБ	35,0 МБ	20,5 кБ	Н/Д	Н/Д	Обычный
termit	vasily	0,00	2682	9,5 МБ	5,1 МБ	Н/Д	Н/Д	Н/Д	Обычный
nm-applet	vasily	0,00	2544	9,2 МБ	3,9 МБ	Н/Д	Н/Д	Н/Д	Обычный
nemo-desktop	vasily	0,00	2543	8,5 МБ	4,6 МБ	Н/Д	Н/Д	Н/Д	Обычный
xapp-sn-watcher	vasily	0,00	2599	7,9 МБ	110,6 кБ	Н/Д	Н/Д	Н/Д	Обычный
csd-background	vasily	0,00	2453	6,3 МБ	1,9 МБ	Н/Д	Н/Д	Н/Д	Обычный
csd-xsettings	vasily	0,00	2454	6,3 МБ	9,1 МБ	Н/Д	Н/Д	Н/Д	Обычный
csd-media-keys	vasily	0,00	2455	6,3 МБ	3,3 МБ	Н/Д	Н/Д	Н/Д	Обычный
polkit-gnome-authentication-agent-1	vasily	0,00	2546	5,5 МБ	94,2 кБ	Н/Д	Н/Д	Н/Д	Обычный
gvfs-udisks2-volume-monitor	vasily	0,00	2565	4,0 МБ	1,4 МБ	Н/Д	Н/Д	Н/Д	Обычный
gvfsd	vasily	0,00	2472	3,0 МБ	3,1 МБ	Н/Д	Н/Д	Н/Д	Обычный
gvfsd-fuse	vasily	0,00	2485	2,9 МБ	356,4 кБ	Н/Д	Н/Д	Н/Д	Обычный
at-spi-bus-launcher	vasily	0,00	2436	2,8 МБ	28,7 кБ	Н/Д	Н/Д	Н/Д	Обычный
bash	vasily	0,00	2689	1,4 МБ	69,6 кБ	Н/Д	Н/Д	Н/Д	Обычный
at-spi2-registryd	vasily	0,00	2444	839,7 кБ	77,8 кБ	Н/Д	Н/Д	Н/Д	Обычный
startx	vasily	0,00	2394	593,9 кБ	2,7 МБ	94,2 кБ	Н/Д	Н/Д	Обычный
gvfsd-metadata	vasily	0,00	2586	573,4 кБ	61,4 кБ	Н/Д	Н/Д	Н/Д	Обычный
dbus-daemon	vasily	0,00	2431	561,2 кБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
dconf-service	vasily	0,00	2630	528,4 кБ	69,6 кБ	Н/Д	Н/Д	Н/Д	Обычный
scrot	vasily	0,00	2696	413,7 кБ	544,8 кБ	Н/Д	Н/Д	Н/Д	Обычный
dbus-daemon	vasily	0,00	2441	372,7 кБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
dbus-launch	vasily	0,00	2430	303,1 кБ	Н/Д	Н/Д	Н/Д	Н/Д	Обычный
xinit	vasily	0,00	2416	151,6 кБ	24,6 кБ	Н/Д	Н/Д	Н/Д	Обычный

# ГЛАВА 11

---

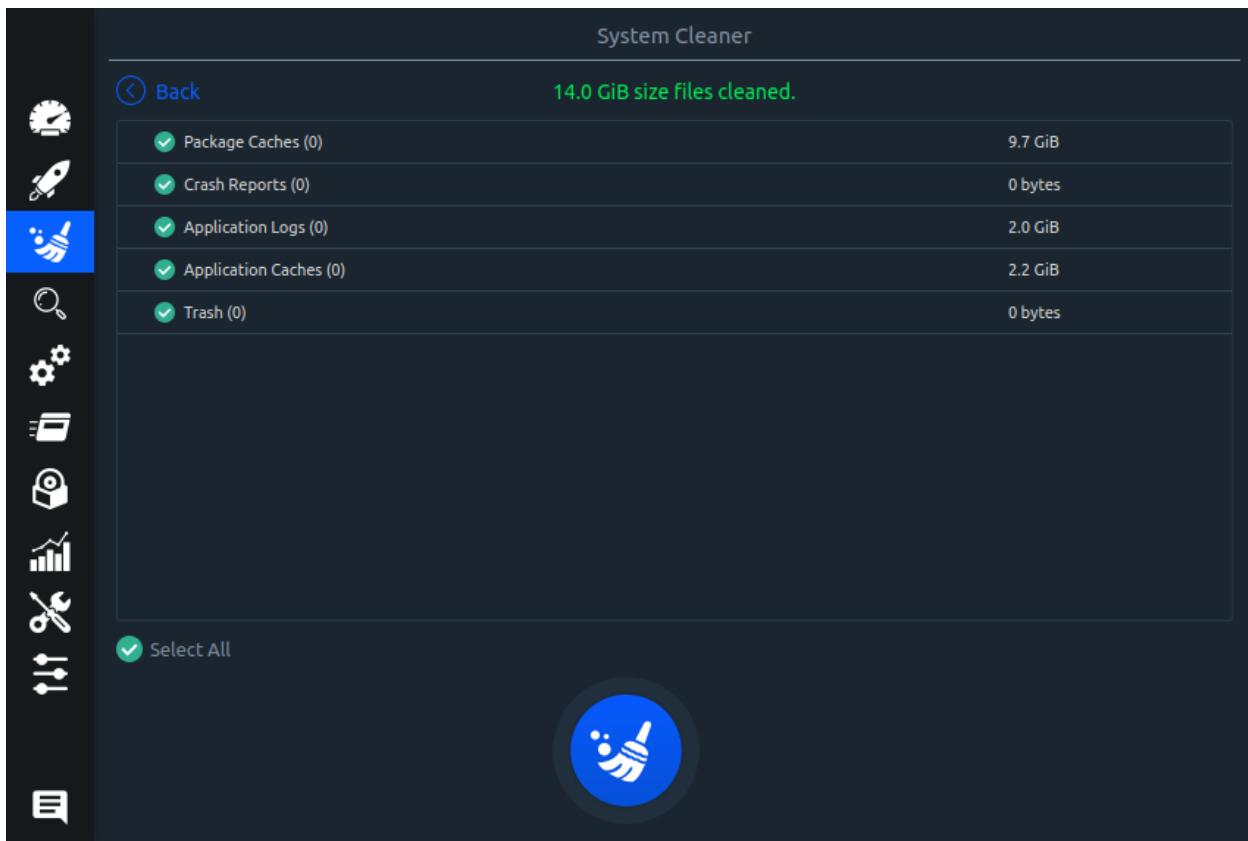
## Полезные программы

---

Программы разного назначения, однако могут быть полезными.

### **11.1 Stacer**

Помощник в обслуживании и чистке системы.



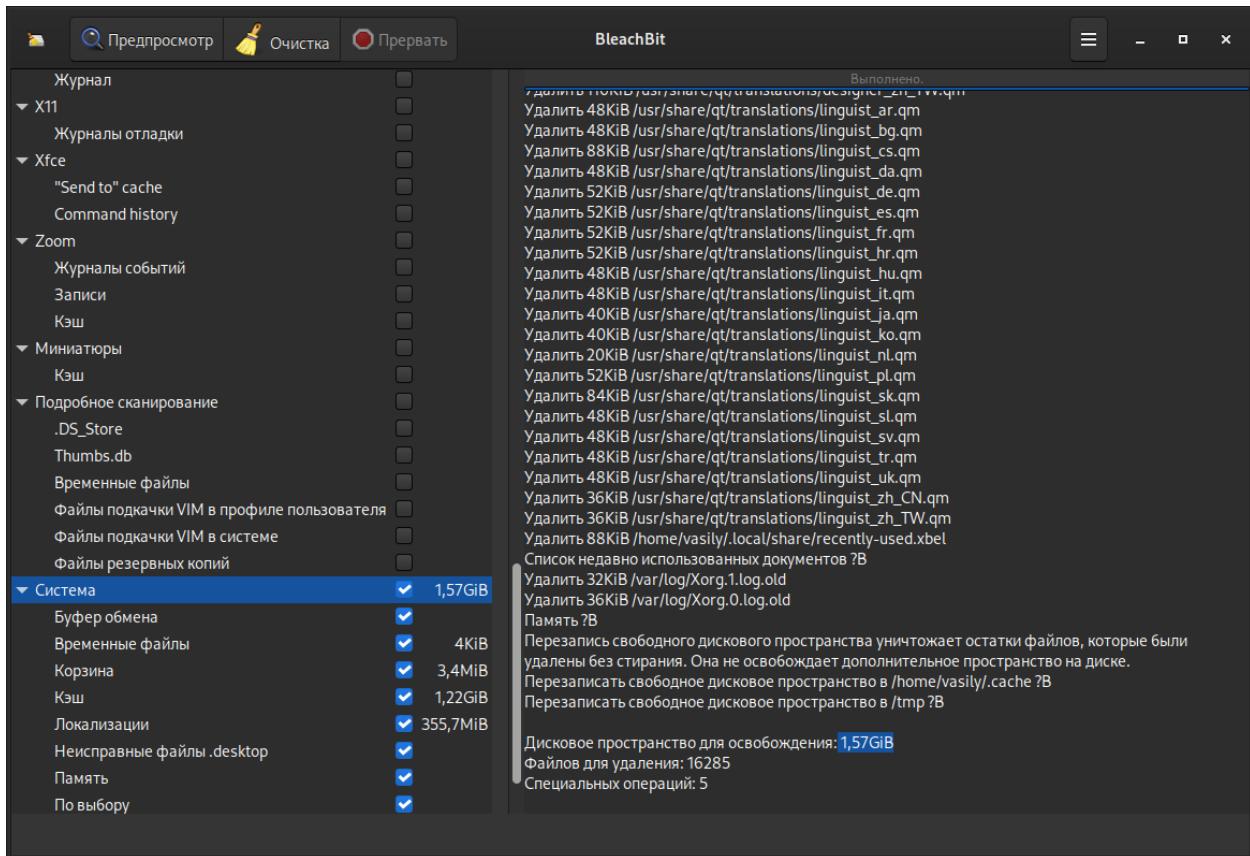
#### Установка:

```
git clone https://aur.archlinux.org/stacer.git # Скачивание исходников.  
cd stacer  
makepkg -src # Переход в stacer.  
makepkg -src # Сборка и установка.
```

## 11.2 Bleachbit

Аналог CCleaner для Linux, помогает выполнить очистку системы от накопившегося в ней мусора.

Советуем выполнять чистку системы уже после проведения всех оптимизаций.



### Установка + дополнительные фильтры:

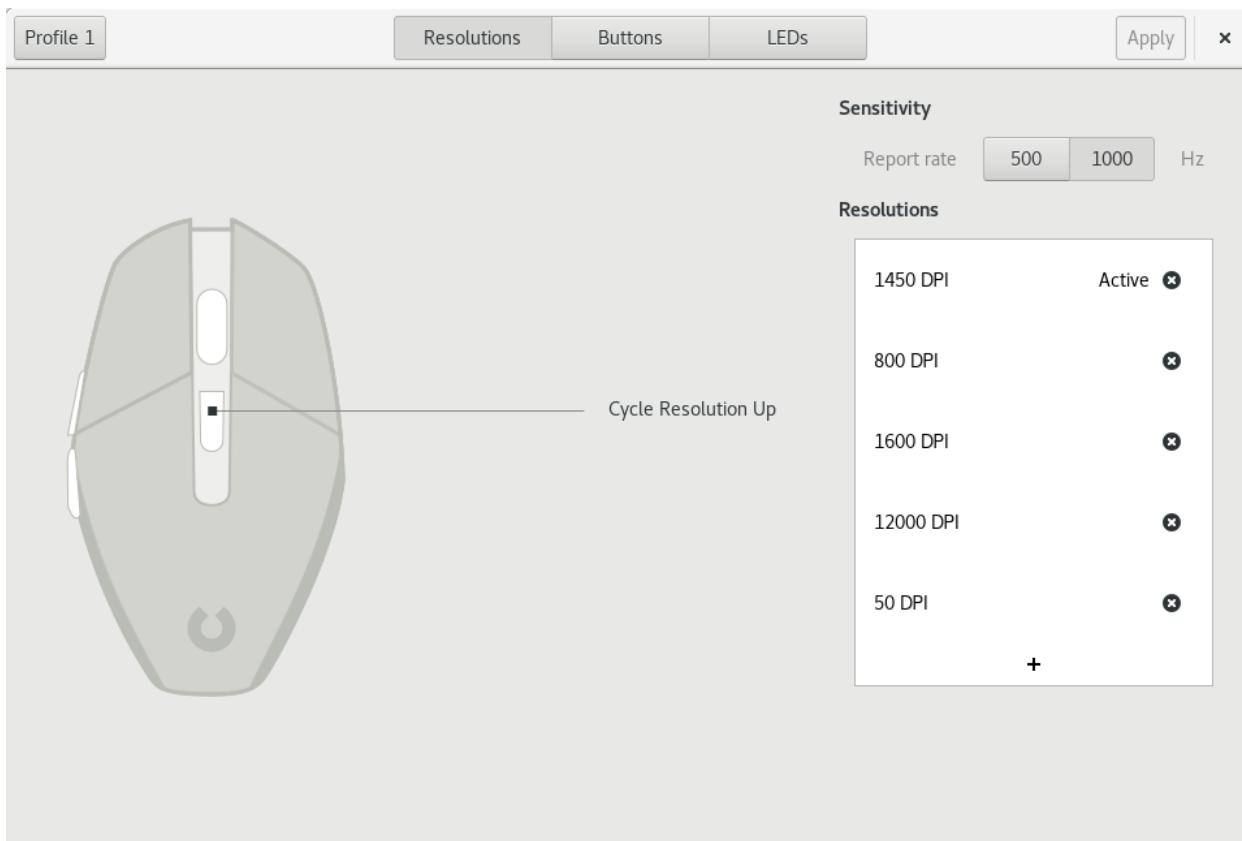
```
sudo pacman -S bleachbit

# Дополнительные фильтры

git clone https://aur.archlinux.org/cleanerml-git.git # Загрузка исходников.
cd cleanerml-git                                     # Переход в cleanerml.
makepkg -srsc                                         # Сборка и установка.
```

## 11.3 Piper

Позволяет выполнить более тонкую настройку вашей мышки, в том числе переназначить DPI, настроить подсветку и собственные действия на дополнительные кнопки.



## Установка

```
sudo pacman -S piper
```

**Внимание:** Поддерживаются только некоторые из моделей мышек от Logitech/Razer/Steelseries. Полный список поддерживаемых устройств вы можете найти по ссылке:

<https://github.com/libratbag/libratbag/wiki/Devices>

## 11.4 pam\_usb

Позволяет сделать из вашей USB-флешки ключ для авторизации в вашу систему. Совместим с экранными менеджерами входа GDM и KDM.

Существует несколько режимов работы:

1. Использовать флешку вместо пароля, при условии её подключения (если подключение отсутствует - нужно вводить пароль)
2. Требовать наличие подключенного USB-носителя вместе с вводом пароля.

## Установка

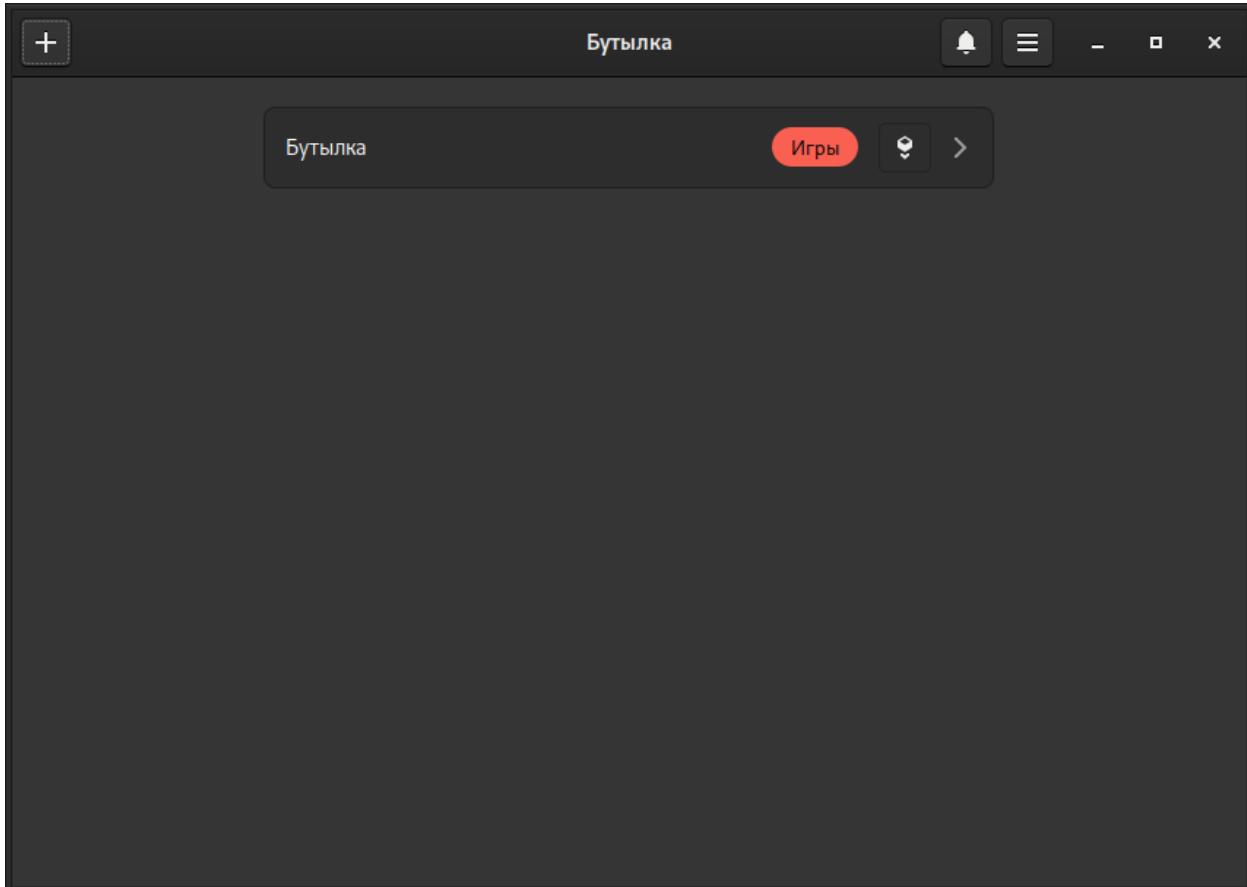
```
git clone https://github.com/mcdope/pam_usb.git  
cd pam_usb/arch_linux  
makepkg -src
```

## 11.5 Bottles

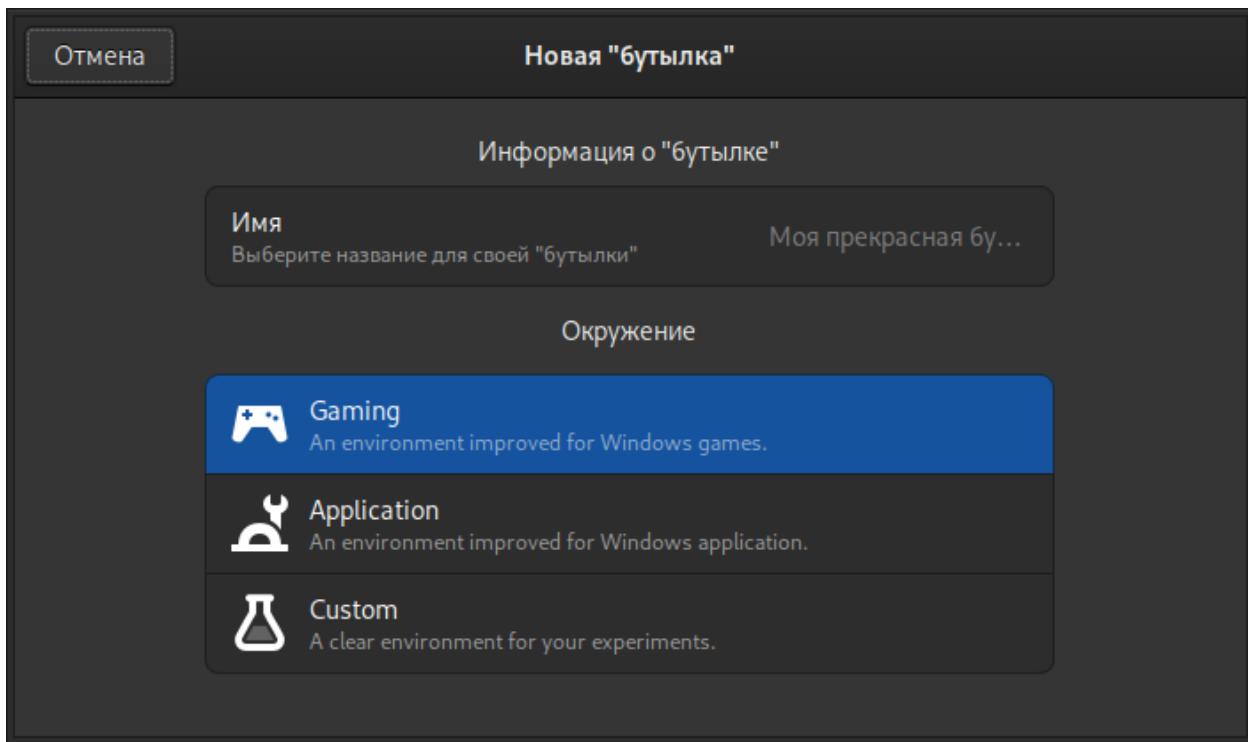
Удобный менеджер по управлению бутылками (префиксами) в Wine. Альтернатива Lutris, имеет приятный и понятный интерфейс, возможность графической установки зависимостей (DLL библиотек) и поддерживает изоляцию из коробки.

### Демонстрация

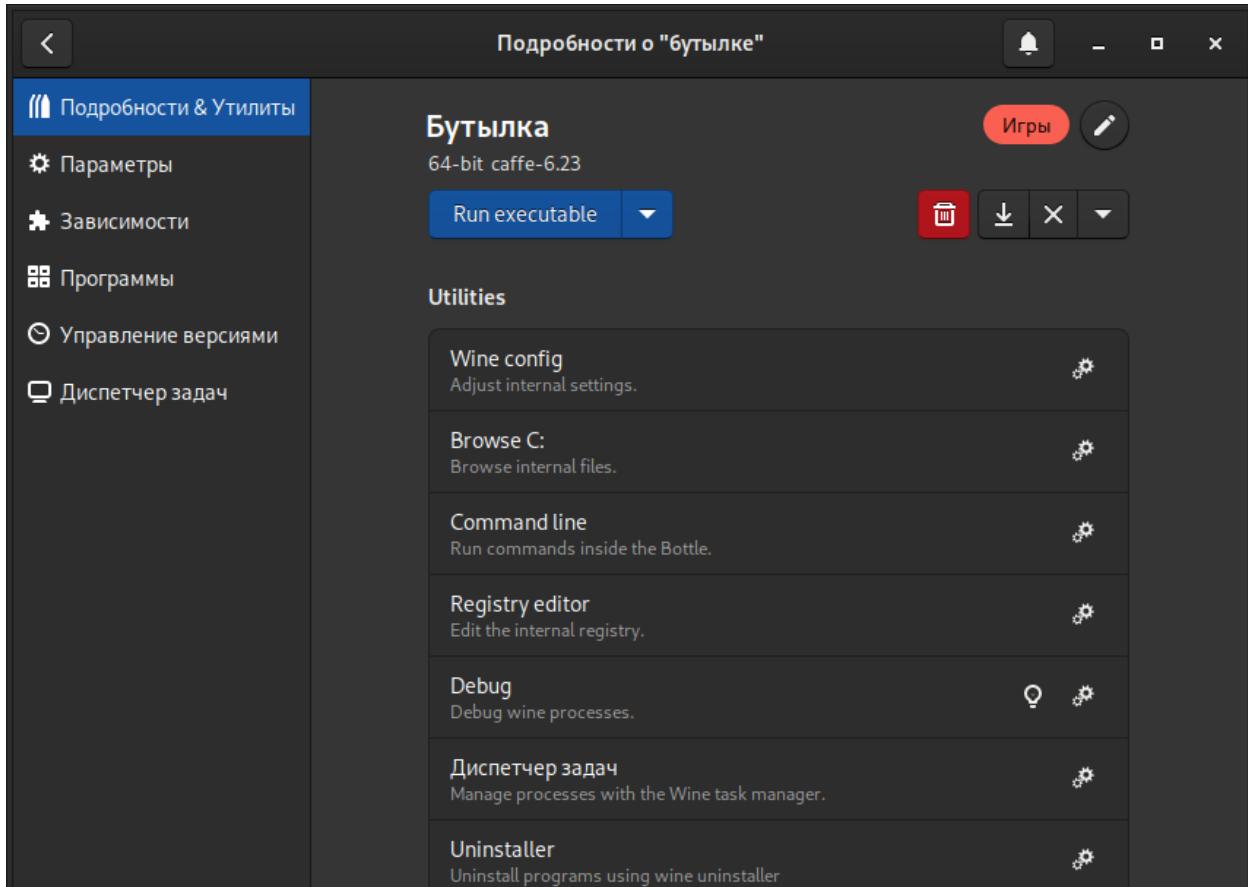
1. Окно выбора бутылки



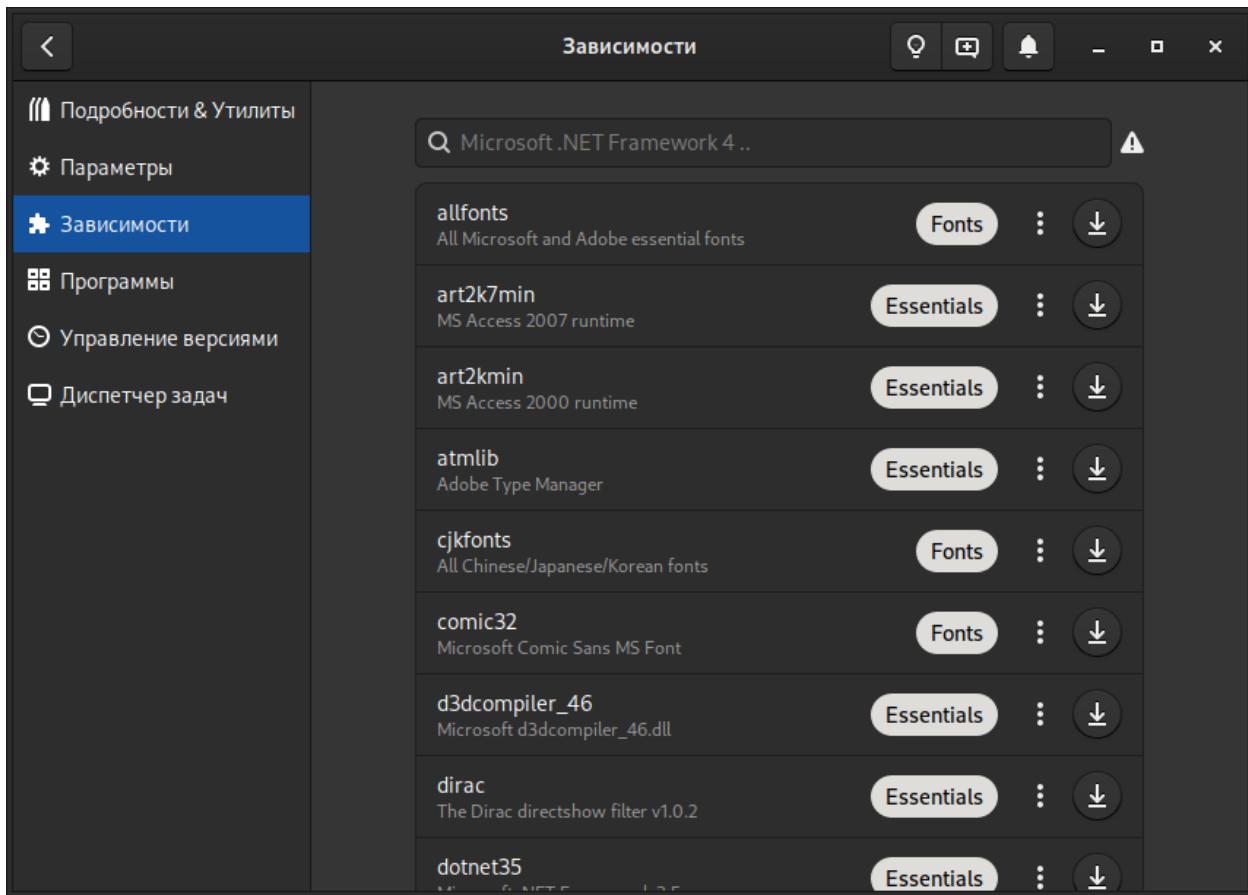
2. Создание новой бутылки



### 3. Управление бутылкой



#### 4. Установка зависимостей (DLL библиотек)



#### Установка

```
git clone https://aur.archlinux.org/bottles.git # Скачиваем исходники
cd bottles
makepkg -src # Переход в директорию
# Сборка и установка
```

---

## Алфавитный указатель

---

### A

about, 77, 81, 85  
alsa, 24  
amd, 5, 7, 12, 26, 27, 94  
ananicy, 22  
async, 83  
audio, 24  
aur, 1  
auto-cpufreq, 31

### B

baloo, 109  
bar, 26  
basic, 4  
basics, 1  
bleachbit, 128  
bootloader, 34  
btrfs, 36, 37, 41, 48

### C

ccache, 21  
cinnamon, 121  
cinnamon-settings-daemon, 121  
clang, 19, 21, 73  
commands, 1  
comperssion, 38  
compositor, 106, 112, 115, 116, 119, 123, 124  
compression, 37, 39, 41, 48  
compsize, 38  
configure, 56  
coredump, 32  
cosmetics, 107  
cpu, 7, 29  
cpupower, 2931

### D

daemons, 104, 109, 110, 118, 121  
dbus-broker, 22

debug, 109  
de-optimizations, 103, 108, 117, 121  
dependencies, 78  
dlss, 97  
driver, 7, 11  
drivers, 5  
dumps, 32  
dxvk, 83, 85, 98, 99

### E

effects, 116, 124  
environment, 11  
ext4, 36

### F

file-indexing, 104, 109  
flags, 19  
fps, 98, 99  
frequencies, 30, 31  
fsr, 94  
fstab, 36

### G

gamemode, 86, 93  
games, 41  
gaming, 77, 78, 81, 8386, 93, 94, 97, 131  
garbage-removal, 104, 108, 118, 127, 128  
gnome, 103, 104, 107  
gnome-control-center, 104  
gnome-settings-daemon, 104  
gnome-shell, 106  
governor, 2931  
gpg, 3  
grub, 34  
grub-customizer, 34  
gui, 30

### H

haveged, 22

hdd, 25  
helpers, 1  
hibernation, 32

**I**

image-scaling, 94, 97  
initramfs, 6  
installation, 4, 5, 22, 24, 78, 80, 81, 83, 84,  
86, 93, 98, 99, 106  
intel, 5, 7, 12

**K**

kde, 108  
kdebugdialog5, 109  
kernel, 32, 49, 50, 52, 54, 56, 73, 99  
key, 3  
kwin, 112, 115, 116

**L**

latency, 11  
linux-tkg, 54  
liquorix, 50  
llvm-bolt-builds, 21  
lowlatency, 24, 83, 112, 115, 116, 119, 123,  
124  
lqx, 50  
lto, 19, 21  
lto native-compilation, 73  
lutris, 86, 93  
lz4, 25  
lzo, 37

**M**

makepkg, 19, 21  
makepkg-conf, 19  
mangohud, 98  
mesa, 12  
microcode, 7  
mini-kernel, 99, 101  
mirrorlist, 4  
mitigations-off, 34, 35  
mkinitcpio, 6, 25  
modprobed-db, 99, 101  
modules, 6, 99, 101  
monitor, 12, 13  
monitoring, 98, 99  
mount, 36  
mouse, 129  
muffin, 123, 124  
multilib, 3  
mutter, 106

**N**

native-compilation, 19, 21, 49, 50, 52, 54,  
78, 80, 81, 83, 84

networkmanager, 25  
no-display-manager, 103  
nohang, 22  
nvidia, 5, 7, 11, 13, 97

**O**

options, 36  
overclocking, 12, 13

**P**

packages, 4  
packaging, 1  
pacman, 1, 3, 4, 26  
parallel-downloading, 26  
patch-off, 34  
performance, 2931  
pgo, 21  
phoronix-test-suite, 39  
pipewire, 24  
plasma, 108110  
plasma-pa, 108  
polkit, 32  
powerpill, 26  
prefixes, 81, 131  
problems, 101  
proton, 81, 93, 97  
proton-ge-custom, 93  
pulseaudio, 24

**R**

reflector, 4  
refresh-rate, 12, 13  
results, 48, 107, 116, 120, 125  
rng-tools, 22

**S**

sam, 26  
security, 130  
service, 25, 104, 110, 118, 121  
services, 104, 109  
settings, 3, 26, 35, 129  
stacer, 127  
startup-acceleration, 25  
steam, 3  
suspend, 32  
swap, 32  
swapfile, 32

**T**

test, 38, 39, 41, 48  
tracker3, 104  
trim, 22  
tweaks, 11, 27

## U

usb, [130](#)  
useful-programs, [127](#)  
userpatches, [80](#)

## V

variables, [11](#)  
vkd3d, [84](#)  
vsync, [112](#), [119](#), [123](#)

## W

wine, [3](#), [77](#), [78](#), [80](#), [81](#), [84](#), [85](#), [131](#)  
wine-builds, [77](#), [78](#), [80](#)  
wine-staging, [78](#)  
wine-tkg, [78](#), [80](#)

## X

x11, [103](#)  
x11-unrediction, [119](#), [123](#)  
x11-unredirection, [115](#)  
xanmod, [52](#)  
xfce, [117](#), [118](#)  
xfce4, [117](#)  
xfwm, [119](#)  
xorg, [7](#)  
xorg-xinit, [103](#)

## Z

zen, [49](#)  
zfs, [36](#)  
zib, [37](#)  
zram, [22](#)  
zstd, [3739](#)