

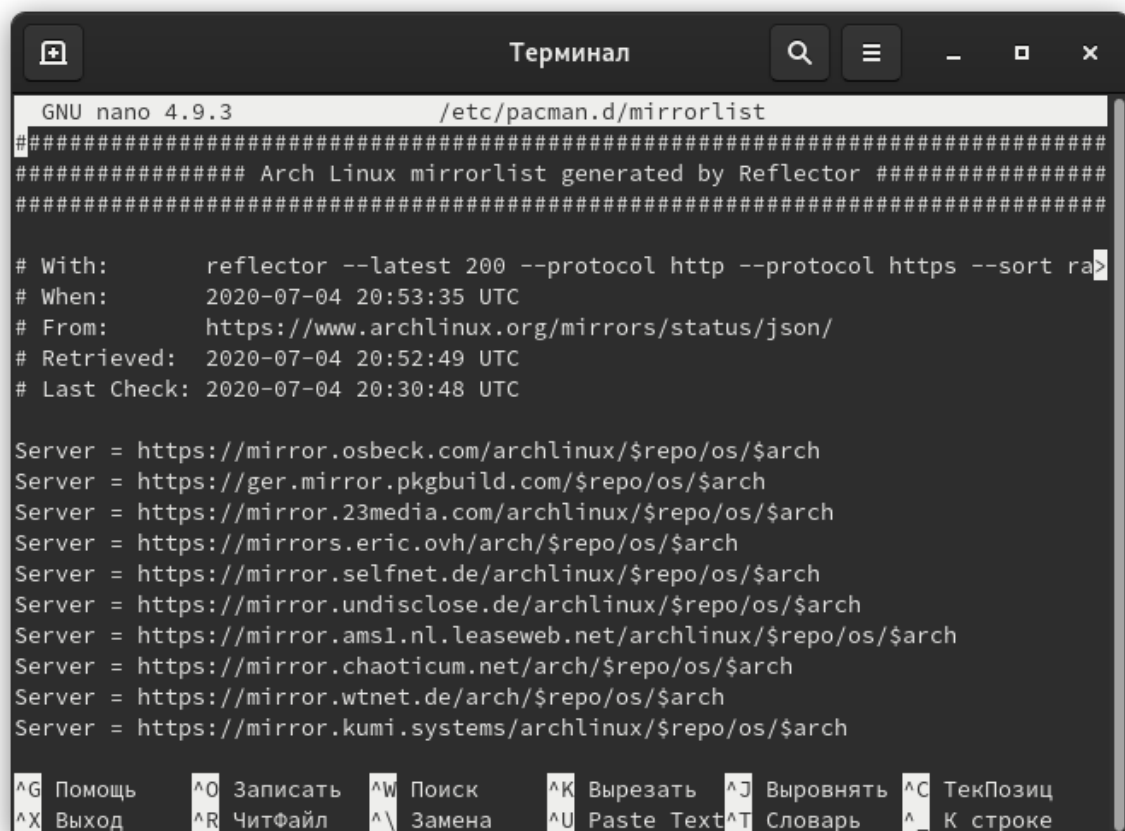
ОТ АВТОРА. (Обновление от 1 сентября).

Данный файл является средством выжить максимум с Archlinux или систем на его основе в плане игровой производительности. Все методы были протестированы автором и являются оптимальными, но как и всегда, все что вы делаете с вашим ПК это ваша ответственность и автор никакой ответственности не несет. Все шаги идут по порядку, старайтесь выполнять аналогично.

1. Получить оптимальное по скорости хранилище. (mirrorlist).

`sudo pacman -S reflector` # Установит штуковину для подсчета.

`sudo reflector --latest 200 --protocol http --protocol https --sort rate --save /etc/pacman.d/mirrorlist` # Выполнит команду для подсчета.



```
GNU nano 4.9.3 /etc/pacman.d/mirrorlist
#####
##### Arch Linux mirrorlist generated by Reflector #####
#####

# With:      reflector --latest 200 --protocol http --protocol https --sort ra>
# When:      2020-07-04 20:53:35 UTC
# From:      https://www.archlinux.org/mirrors/status/json/
# Retrieved: 2020-07-04 20:52:49 UTC
# Last Check: 2020-07-04 20:30:48 UTC

Server = https://mirror.osbeck.com/archlinux/$repo/os/$arch
Server = https://ger.mirror.pkgbuild.com/$repo/os/$arch
Server = https://mirror.23media.com/archlinux/$repo/os/$arch
Server = https://mirrors.eric.ovh/arch/$repo/os/$arch
Server = https://mirror.selfnet.de/archlinux/$repo/os/$arch
Server = https://mirror.undisclose.de/archlinux/$repo/os/$arch
Server = https://mirror.amsl.nl.leaseweb.net/archlinux/$repo/os/$arch
Server = https://mirror.chaoticum.net/arch/$repo/os/$arch
Server = https://mirror.wtnet.de/arch/$repo/os/$arch
Server = https://mirror.kumi.systems/archlinux/$repo/os/$arch

^G Помощь  ^O Записать  ^W Поиск    ^K Вырезать  ^J Выводить  ^C ТекПозиц
^X Выход   ^R ЧитФайл  ^\ Замена  ^U Paste Text ^T Словарь  ^_ К строке
```

Вы можете вручную добавить Русское зеркало, просто отредактировав через любой редактор (`/etc/pacman.d/mirrorlist`).

Пример команды редактирования через редактор nano.

`sudo nano /etc/pacman.d/mirrorlist` # Снять решетку с ссылки, страну не трогать, поставить её на первое место по списку.

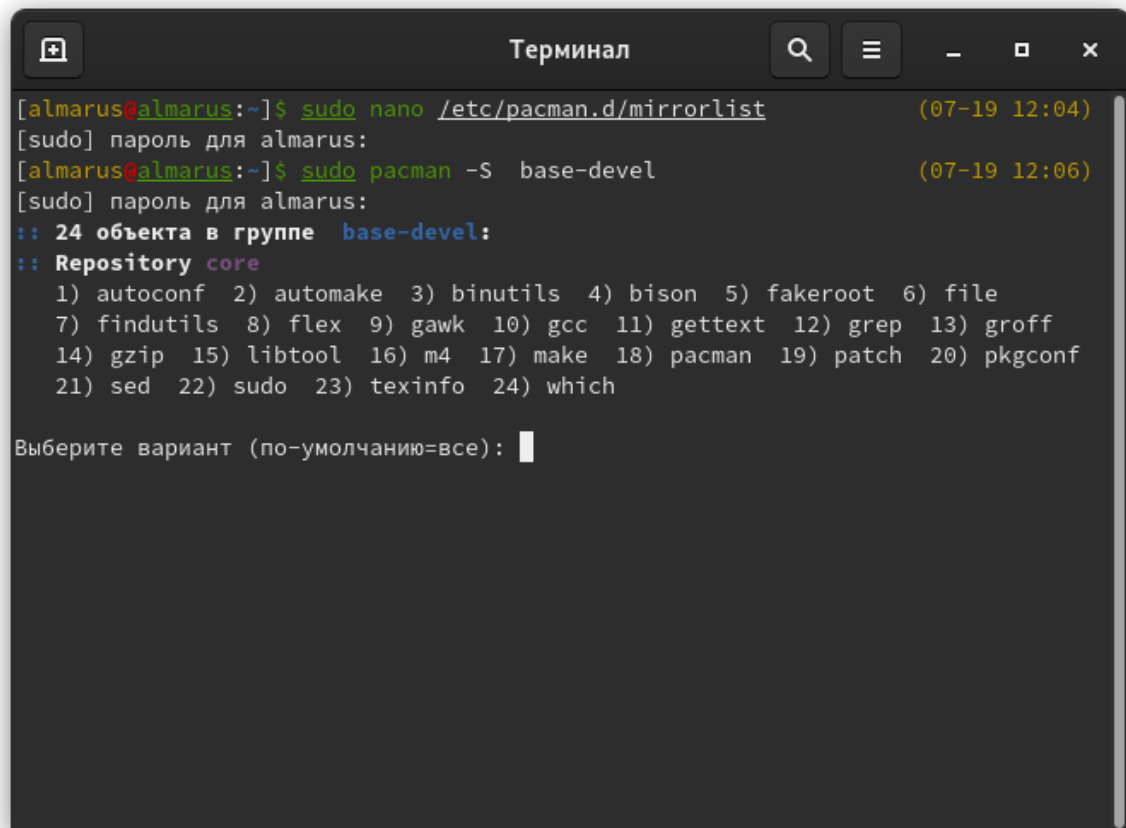
Всегда свежие листы.

<https://www.archlinux.org/mirrorlist/>

1.1 Обновить систему.

1.2 Установить базовые пакеты для сборки программ.

`sudo pacman -S base-devel` # Пакет для сборки программ.



```
[almarus@almarus:~]$ sudo nano /etc/pacman.d/mirrorlist (07-19 12:04)
[sudo] пароль для almarus:
[almarus@almarus:~]$ sudo pacman -S base-devel (07-19 12:06)
[sudo] пароль для almarus:
:: 24 объекта в группе base-devel:
:: Repository core
   1) autoconf  2) automake  3) binutils  4) bison  5) fakeroot  6) file
   7) findutils 8) flex      9) gawk    10) gcc   11) gettext 12) grep  13) groff
  14) gzip     15) libtool  16) m4    17) make  18) pacman 19) patch 20) pkgconf
  21) sed      22) sudo    23) texinfo 24) which

Выберите вариант (по-умолчанию=все):
```

1.3 Установить редакторы и помощники.

`sudo pacman -S nano` # Простой консольный редактор, аналог gedit.

`sudo pacman -S git` # Нужен для загрузки через консоль.

2.0 Игровое ядро.

Ядро в Linux-Unix подобных системах это фундамент, что отвечает за все операции и каково будет его состояние, такова будет производительность системы. По умолчанию в Linux-ядре используются настройки ориентированные на работу с серверами и сохранении энергии, что нам не нужно.

Вариант 1.

`sudo pacman -S linux-zen linux-zen-headers` # Данная команда выполнит установку не оригинального ядра, что уже настроена и собрана под системы для хорошей производительности.

Минусы - Собрано для всех систем, а значит не все выжали.

Плюсы - Простая установка, не нужно собирать самому, обновляется.

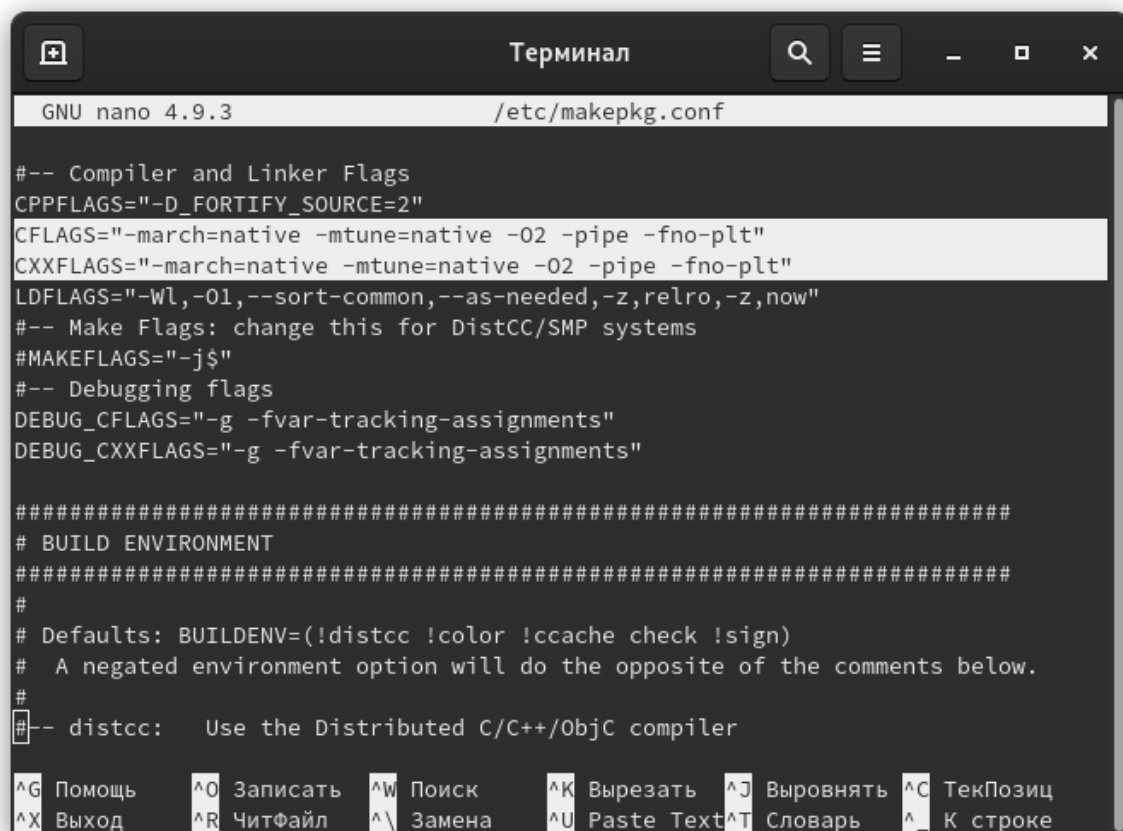
Вариант 2 # Обновление от 20 июля - странная вещь, почему-то включаются технологии сохранения энергии (сброс частоты, что вызывает фризы), воздержитесь от использования данного ядра, до обновления). Присмотритесь к [lqx](#), особенно если у вас Windows.

<https://aur.archlinux.org/packages/linux-xanmod/>

cd tools # Данный параметр заставляет вас перейти в папку tools , что была создана вами заранее (создайте) в домашнем каталоге.

git clone <https://aur.archlinux.org/linux-xanmod.git> # Скачать файлы для сборки.

sudo nano /etc/makepkg.conf # Редактирование позволит собирать пакеты под наш ПК с максимальной отдачей.



```
GNU nano 4.9.3 /etc/makepkg.conf

#-- Compiler and Linker Flags
CPPFLAGS="-D_FORTIFY_SOURCE=2"
CFLAGS="-march=native -mtune=native -O2 -pipe -fno-plt"
CXXFLAGS="-march=native -mtune=native -O2 -pipe -fno-plt"
LDFLAGS="-Wl,-O1,--sort-common,--as-needed,-z,relro,-z,now"
#-- Make Flags: change this for DistCC/SMP systems
#MAKEFLAGS="-j$"
#-- Debugging flags
DEBUG_CFLAGS="-g -fvar-tracking-assignments"
DEBUG_CXXFLAGS="-g -fvar-tracking-assignments"

#####
# BUILD ENVIRONMENT
#####
#
# Defaults: BUILDDIR=(!distcc !color !ccache check !sign)
# A negated environment option will do the opposite of the comments below.
#
#-- distcc: Use the Distributed C/C++/ObjC compiler

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Вывести ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^_ К строке
```

Перевести параметры **march** и **mtune** на **native** - сохранить (**контрл+x**) (**y,enter**).

Самыми оптимальными флагами для оптимизации ядра я считаю:

> CFLAGS= -march=native -mtune=native -O3 -ffast-math -flto -funroll-loops -mfpmath=sse

Аналогично для CXXFLAGS.

ls # Команда покажет файлы в папку tools

cd linux-xanmod # Перейдем в папку linux-xanmod

sudo nano /home/almarus/tools/linux-xanmod/PKGBUILD

Изменить значения в этих строчках на данные.

microarchitecture=42

```
use_numa=n
use_tracers=n
use_pds=n
use_ns=n
Сохранить.
makepkg -si # Начать установку
```

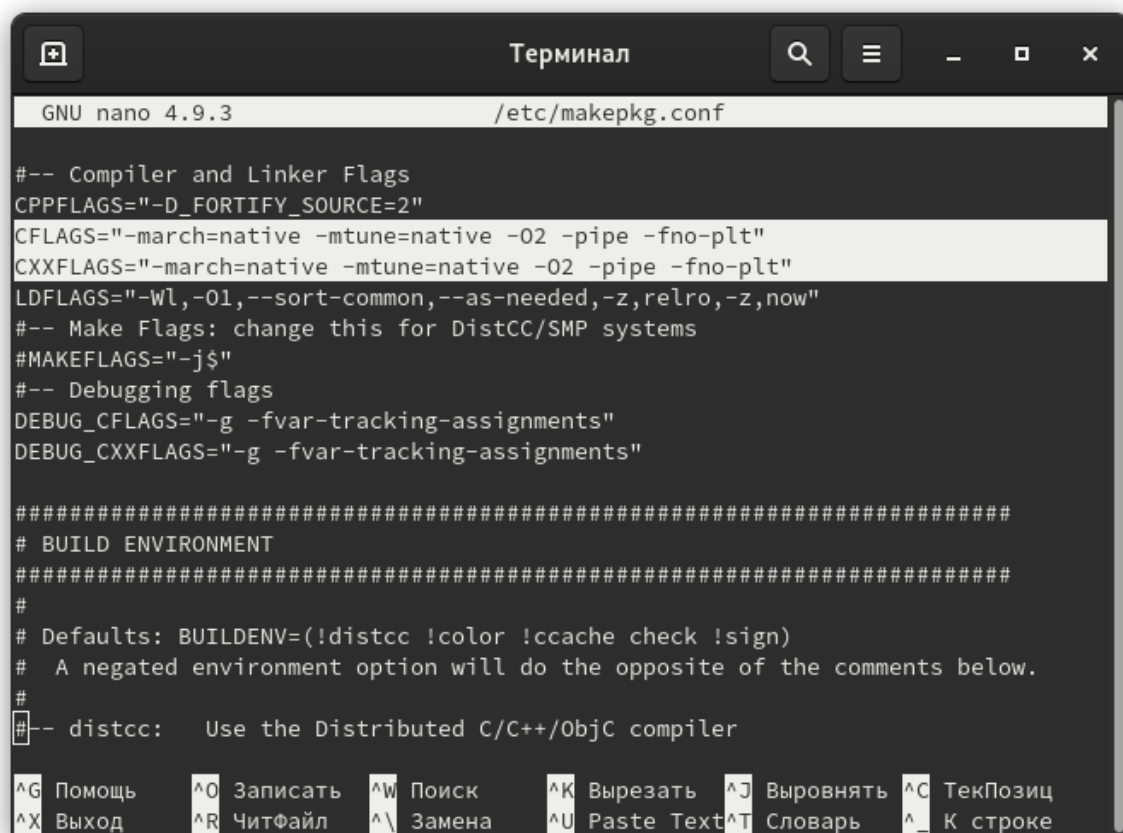
Вариант 3 # Компиляция ядра с графическим интерфейсом.

<https://aur.archlinux.org/packages/linux-lqx/>

cd tools # Данный параметр заставляет вас перейти в папку tools , что была создана вами заранее (создайте) в домашнем каталоге.

git clone https://aur.archlinux.org/linux-lqx.git # Скачать файлы для сборки.

sudo nano /etc/makepkg.conf # Редактирование позволит собирать пакеты под наш ПК с максимальной отдачей.



```
GNU nano 4.9.3 /etc/makepkg.conf

#-- Compiler and Linker Flags
CPPFLAGS="-D_FORTIFY_SOURCE=2"
CFLAGS="-march=native -mtune=native -O2 -pipe -fno-plt"
CXXFLAGS="-march=native -mtune=native -O2 -pipe -fno-plt"
LDFLAGS="-Wl,-O1,--sort-common,--as-needed,-z,relro,-z,now"
#-- Make Flags: change this for DistCC/SMP systems
#MAKEFLAGS="-j$"
#-- Debugging flags
DEBUG_CFLAGS="-g -fvar-tracking-assignments"
DEBUG_CXXFLAGS="-g -fvar-tracking-assignments"

#####
# BUILD ENVIRONMENT
#####
#
# Defaults: BUILDENV=(!distcc !color !ccache check !sign)
# A negated environment option will do the opposite of the comments below.
#
#-- distcc: Use the Distributed C/C++/ObjC compiler

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выводить ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^_ К строке
```

Перевести параметры **march** и **mtune** на **native** - сохранить (контрл+x) (y,enter).

ls # Команда покажет файлы в папку tools

cd linux-lqx # Перейдем в папку linux-lqx

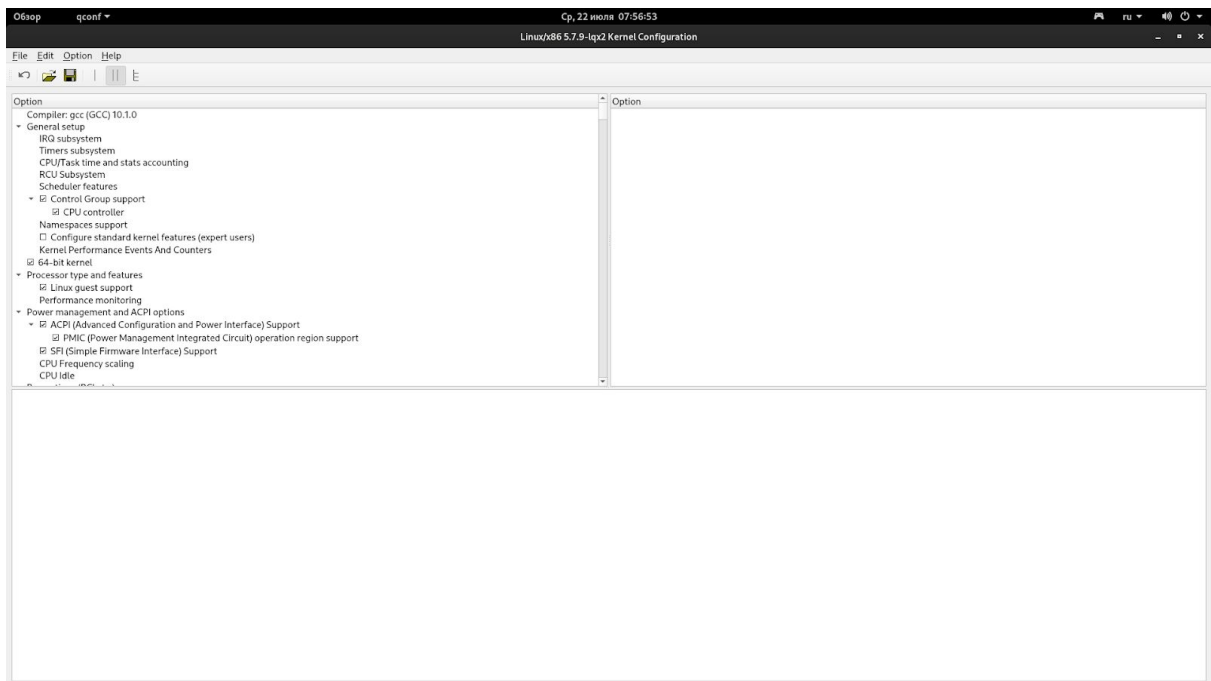
sudo nano /home/almarus/tools/linux-lqx/PKGBUILD

Изменить значения в этих строках на данные.

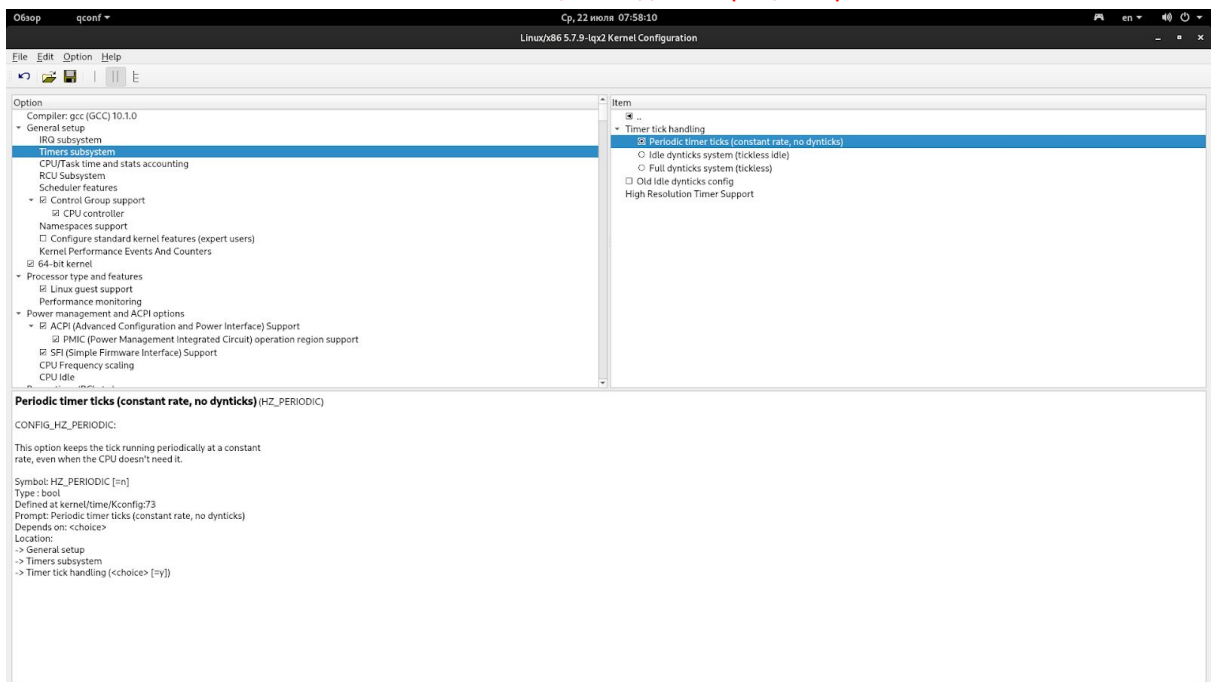
_makepkg=y

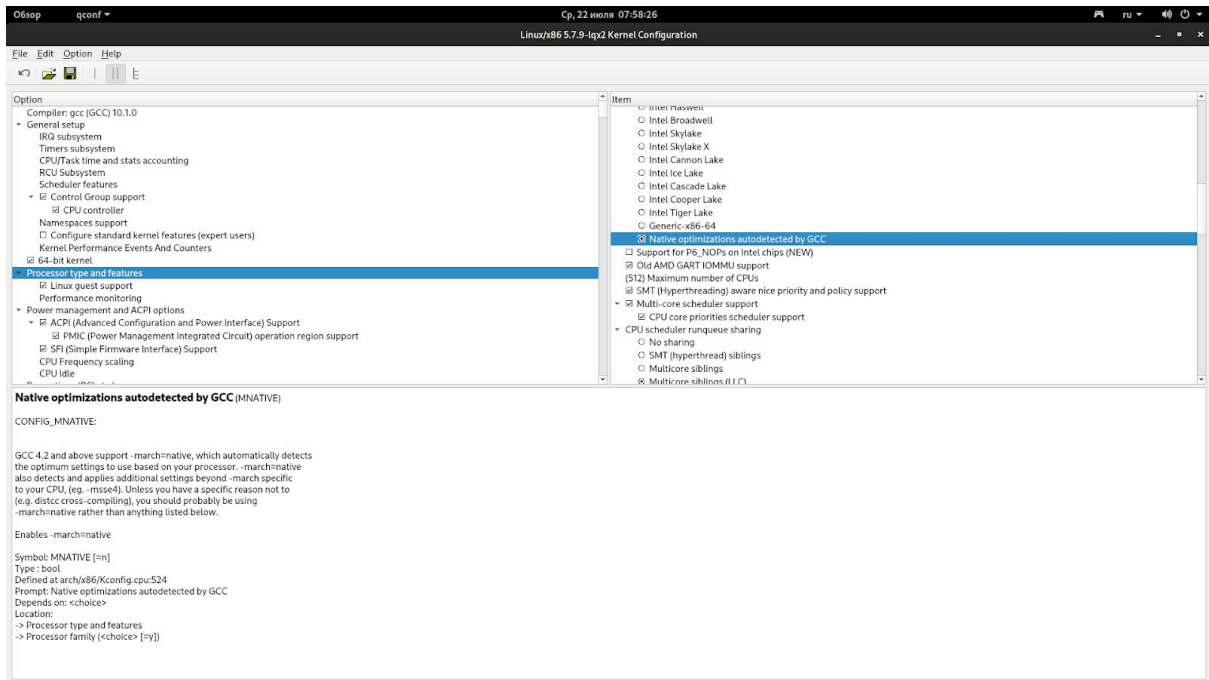
Сохранить.

makepkg -si # Начать установку



В появившемся окне выполняем оптимизацию под сам процессор



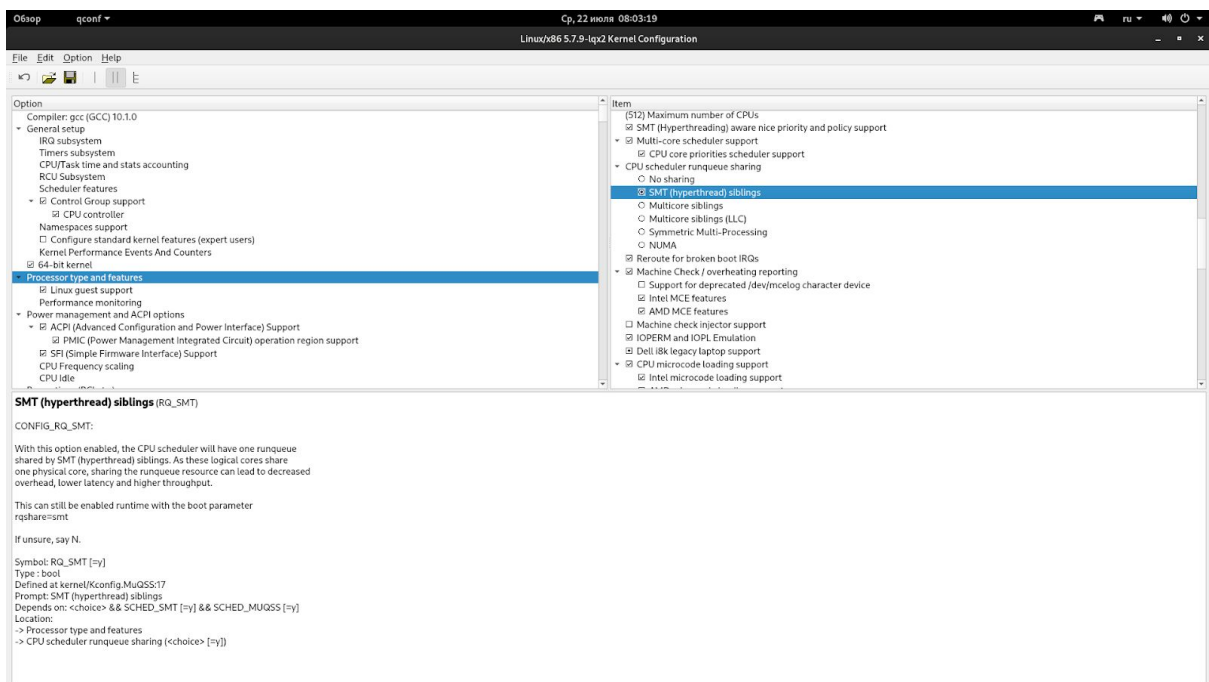


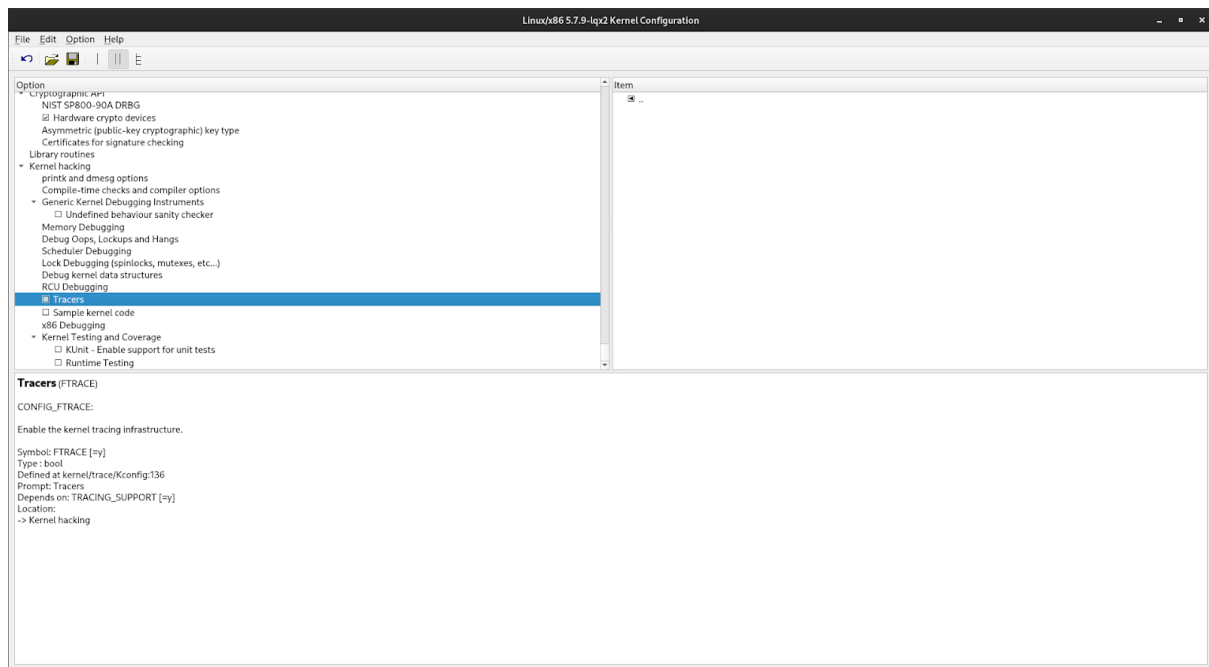
тут выбираем с учетом нашего процессора # Я тестировал с g4560 smt и multicore LLC , со 2 производительность (часто на глаз мне кажется выше).

STM - все 4-8 поточные процессоры

Multicore - процессоры без гибертрейдинга (чистые ядра)

Multicore LLC - райзены и ксеоны





Вариант 4 # Минимально ядро игровое ядро.

Внимание, данное ядро это результат мучений на протяжении 2 дней, но оно того стоило. Результат это сокращение времени компиляции ядра, времени загрузки, размера жерства места на жестком диске и оперативной памяти, с увеличением отклика программ. Минусы в том, что невозможно использовать новое оборудование и порты без перекомпиляции ядра.

1. Устанавливаем <https://aur.archlinux.org/packages/modprobed-db/> по аналогии с hanmod, он нужен для индексации наших модулей.

Запускаем сервис слежения **systemctl --user enable --now modprobed-db.service**

Выключаем и включаем пк пару раз, подключаем все что хотим использовать (флешки и тп).

Вводим по порядку.

sudo modprobed-db recall # Проиндексирует модули и добавит необходимые.

modprobed-db recall # Еще раз

Выполняем редактирование и компиляцию ядра аналогично с примером 3 (тот что выше), но дополнительно добавив **_localmodcfg=y** в **PKGBUILD**.

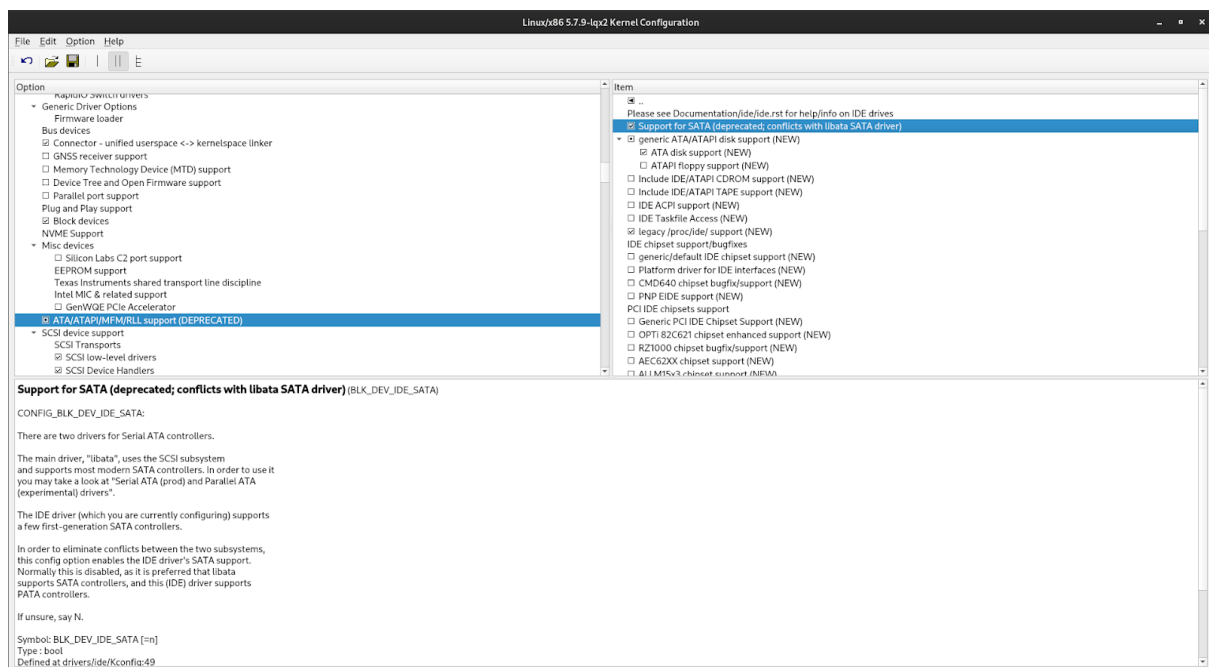
Выполняем компиляцию, на все вопросы отвечаем да, и продолжаем

оптимизацию под процессор на примере варианта 3, но добавив поддержку SATA - по дефолту будет снято - если что-то не понятно стучи в вк

<https://vk.com/ustavchiy>.

Если ошибка

ERROR: device 'UUID=df76929e-ab31-41fc-9271-f89f6aeb58f3' not found. Skipping fsck. (ред.)Mount: /new_root: can't find UUID=df76929e-ab31-41fc-9271-f89f6aeb58f3.



Если при сборке уже скомпилированного ядра выдаст ошибку на отсутствующие модули, что-то формата db_XXX, bd_XXX - просто добавьте их в ручную.
sudo nano /home/ваше имя пользователя/.config/modprobed.db

sudo modprobed-db recall
modprobed-db recall

Модули будут успешно добавлены - можно заново начинать компиляцию.

Данный вариант установки оптимизирует ядро именно под ваш процессор, что в конечном итоге позволит добиться +20% производительности из воздуха.
 Но установка может потребовать много времени, все зависит от процессора.
 После установки выполнить по порядку.

sudo pacman -S intel-ucode iucode-tool (Intel) # Установить микрокод
sudo pacman -S amd-ucode iucode-tool (AMD) # Установить микрокод
sudo grub-mkconfig -o /boot/grub/grub.cfg # Обновить загрузчик.

Видеоверсия

<https://www.youtube.com/watch?v=8GRNN94afyQ>

Сбой, неизвестный ключ. # Может встретиться при установке ядра.

gpg --keyserver keys.gnupg.net --recv-keys ваш ключ.

2.2 Установка lutris и видеодрайверов (на любую видеокарту)

`sudo pacman -S lutris` # Комбайн, где содержатся все удобные настройки.

При запуске lutris без установленных драйверов он начнет ругаться и перенаправит вас на страницу с ними. (Синяя ссылка).

<https://github.com/lutris/docs/blob/master/InstallingDrivers.md>

Nvidia

```
sudo pacman -S nvidia-dkms nvidia-utils lib32-nvidia-utils nvidia-settings
vulkan-icd-loader lib32-vulkan-icd-loader
```

AMD

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

Intel

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

Установить Wine и доп 32 зависимости и доп пакеты для opengl. (Запуск вин программ под линукс).

```
sudo pacman -S wine-staging giflib lib32-giflib libpng lib32-libpng libldap lib32-libldap
gnutls lib32-gnutls mpg123 lib32-mpg123 opengl lib32-opengl v4l-utils lib32-v4l-utils
libpulse lib32-libpulse libgpg-error lib32-libgpg-error alsa-plugins lib32-alsa-plugins
alsa-lib lib32-alsa-lib libjpeg-turbo lib32-libjpeg-turbo sqlite lib32-sqlite libxcomposite
lib32-libxcomposite libxinerama lib32-libgcrypt libgcrypt lib32-libxinerama ncurses
lib32-ncurses opengl-icd-loader lib32-opengl-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
```

3.0 Установка браузера, видеоплеера, steam, ccleaner.

```
sudo pacman -S chromium vlc steam bleachbit
```

Отключение подкачки

```
sudo swapoff /dev/sdxy # Вместо xy ваше название (пример sdb1).
```

```
sudo swapoff -a  
sudo rm -f /swapfile  
sudo nano /etc/fstab # Уберите самую нижнюю строчку полностью.
```

Zramswap

<https://aur.archlinux.org/packages/zramswap/>

Установите по аналогии с linux-xanmod.

Запустите после установки службу

```
sudo systemctl enable zramswap
```

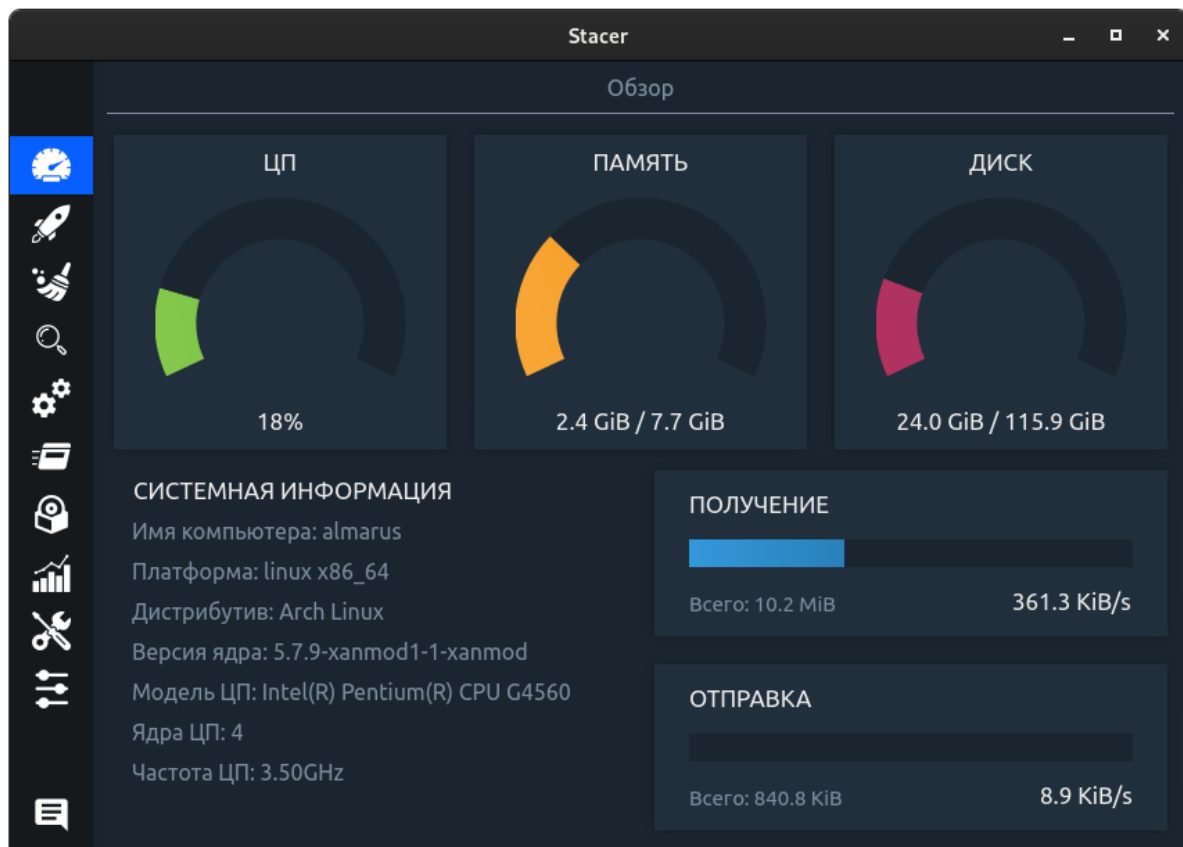
Irqbalance # Устарел, не требуется для ядер выше 4.9.

https://www.archlinux.org/packages/extra/x86_64/irqbalance/

Stacer # Чистилка + помощник.

<https://aur.archlinux.org/packages/stacer/>

Установите по аналогии с linux-xanmod.



Ananicy # Очень ускоряет систему

<https://aur.archlinux.org/packages/ananicy/>

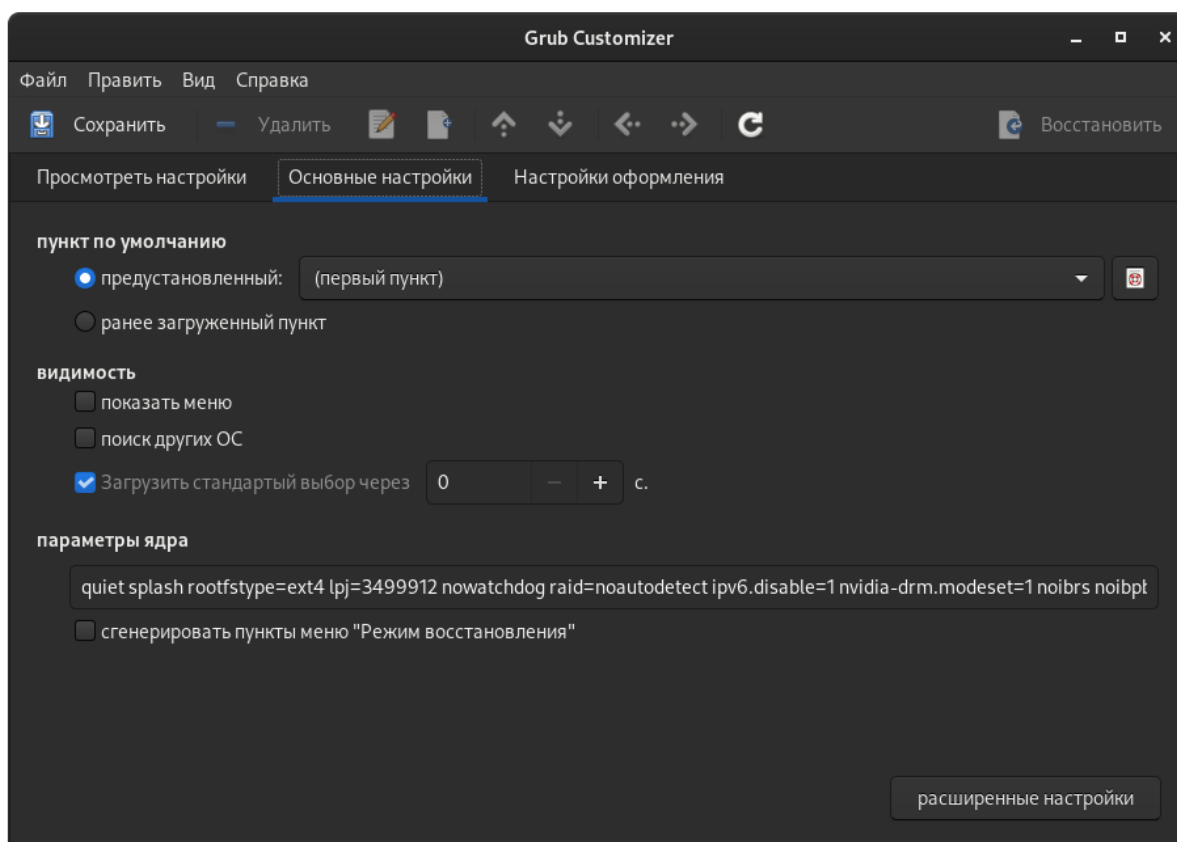
Установите по аналогии с linux-xanmod.

Запустите после установки службу

```
sudo systemctl enable ananicy
```

Grub-update

`sudo pacman -S grub-customizer` # Графическая утилита для обновления grub



Отключение фиксов, что снижают фпс. # Ввести данные как на скриншоте выше, дважды сохранить (правый угол) в 2 окошках 1-2.

```
quiet splash noibrs noibpb nopti nospectre_v2 nospectre_v1 l1tf=off
nospec_store_bypass_disable no_stf_barrier mds=off tsx=on tsx_async_abort=off
rootfstype=ext4 lpj=поменять на свой nowatchdog raid=noautodetect ipv6.disable=1
nvidia-drm.modeset=1 mitigations=off
```

Узнать свой

`dmesg | grep 'lpj='`

3.1 Перевод процессора из стандартного электросбережения в режим производительности.

`sudo pacman -S cpupower` # Установит демона - службу , что будет выставлять.

`sudo cpupower frequency-set -g performance` # Выставит производительность до перезагрузки.

`sudo nano /etc/systemd/system/cpupower.service`

Вставьте туда # Возможно устарело - используйте графический аналог.

[Unit]

Description=Set CPU governor to performance

[Service]

Type=oneshot

ExecStart=/usr/bin/cpupower -c all frequency-set -g performance

[Install]

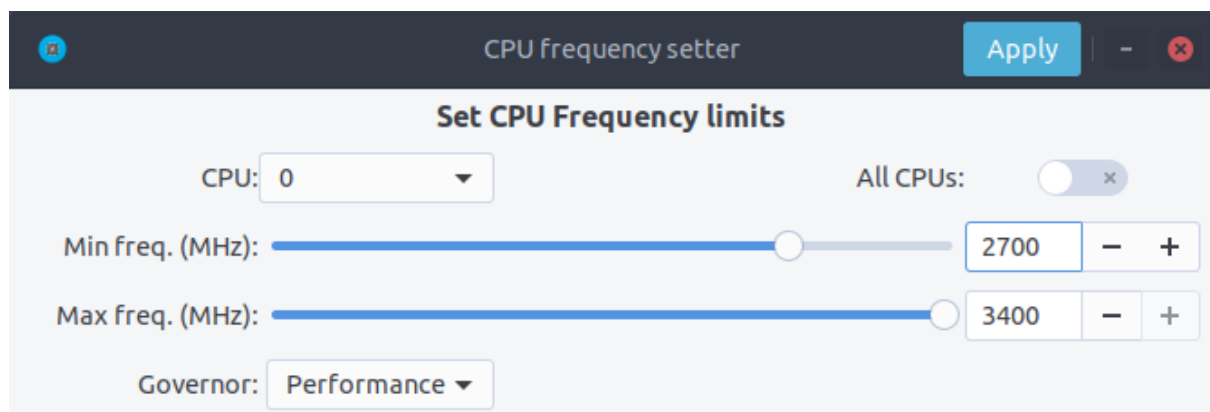
WantedBy=multi-user.target

`sudo systemctl enable cpupower.service` # Запустит как постоянную службу, что установит вычный performance.

Графический аналог # Не работает с ханмод.

Установите по аналогии с linux-xanmod.

<https://aur.archlinux.org/packages/cpupower-gui>



3.2 Trim SSD

`sudo systemctl enable fstrim.timer`

или вручную `sudo fstrim -v /`

3.3 Разгон монитора. # Только Nvidia.

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

3.4 Установка Android

<https://www.youtube.com/watch?v=fgCrsA788KE>

3.5 Ускорить загрузку # Жесткий диск

Убедитесь, что lz4 установлено.

sudo nano /etc/mkinitcpio.conf

- Добавить lz4 lz4_compress в MODULES список (ограничен ())
- Раскомментируйте или добавьте строку с надписью COMPRESSION="lz4"
- Добавьте строку, говоря COMPRESSION_OPTIONS="-9"
- Добавить shutdown в HOOKS список (ограничен ())

3.6 Включить мониторинг в играх

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

3.7 Игровой мод

<https://www.youtube.com/watch?v=RFSI0s-iLWs>

3.8 Nvidia DRM и фикс фризов с nvidia.

sudo nano /etc/mkinitcpio.conf

Добавь это как на скрине и выполни команды ниже
MODULES=()



```
GNU nano 4.9.3 /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=(sis 60-disk nls)
MODULES=(nvidia nvidia_modeset nvidia_uvm nvidia_drm)

# BINARIES
# This setting includes any additional binaries a given user may
# wish into the CPIO image. This is run last, so it may be used to
# override the actual binaries included by a given hook
# BINARIES are dependency parsed, so you may safely ignore libraries
BINARIES=()

# FILES
# This setting is similar to BINARIES above, however, files are added
# as-is and are not parsed in any way. This is useful for config files.
FILES=()

# HOOKS
# This is the most important setting in this file. The HOOKS control the
# modules and scripts added to the image, and what happens at boot time.
# Order is important, and it is recommended that you do not change the
# order in which HOOKS are added. Run 'mkinitcpio -H <hook name>' for
# help on a given hook.
# 'base' is _required_ unless you know precisely what you are doing.
# 'udev' is _required_ in order to automatically load modules
# 'filesystems' is _required_ unless you specify your fs modules in MODULES
# Examples:
# This setup specifies all modules in the MODULES setting above.
# No raid, lvm2, or encrypted root is needed.
# HOOKS="base udev autodetect keyboard keymap consolefont modconf block lvm2 filesystems fsck"
#
# This setup will autodetect all modules for your system and should
# work as a same default.
# HOOKS="base udev autodetect keyboard keymap consolefont modconf block lvm2 filesystems fsck"
#
# This setup will generate a 'full' image which supports most systems.
# No autodetection is done.
# HOOKS="base udev autodetect keyboard keymap consolefont modconf block lvm2 filesystems fsck"
#
# This setup assembles a pata mdadm array with an encrypted root FS.
# Note: See 'mkinitcpio -H mdadm' for more information on raid devices.
# HOOKS="base udev autodetect keyboard keymap consolefont modconf block lvm2 filesystems fsck"
#
# This setup loads an lvm2 volume group on a usb device.
# HOOKS="base udev autodetect keyboard keymap consolefont modconf block lvm2 filesystems fsck"
#
# NOTE: If you have /usr on a separate partition, you MUST include the
# usr, fsck and shutdown hooks.
```

sudo mkinitcpio -p linux-zen # LINUX-zen это название ядра, может отличаться.

3.9 Повышает производительность путем лучшей обработки памяти.

<https://aur.archlinux.org/packages/nohang-git/>

Установите по аналогии с linux-xanmod.

Запустите после установки службу

```
sudo systemctl enable --now nohang-desktop
```

4.0 Поточковая оптимизация (только для Nvidia с проприетарным драйвером).

Nvidia - лучший друг Линуксоида, когда речь заходит о 3D и играх. И начиная с драйвера версии 310, в нём появилась возможность серьёзно увеличить производительность, включив многопоточный рендеринг. По умолчанию данная опция не включена, так как есть приложения, которые не работают с ней (например игра Metro Last Light). Включить эту опцию можно следующими способами:

Вручную: в терминале командой `export __GL_THREADED_OPTIMIZATIONS=1;`

Автоматически при запуске системы: открыть файл `/etc/profile` и в конец вставить строку `__GL_THREADED_OPTIMIZATIONS=1;`

Запускать приложение с этим параметром: например `__GL_THREADED_OPTIMIZATIONS=1 steam`, либо в самом Steam указать данную опцию в параметрах запуска игры: `__GL_THREADED_OPTIMIZATIONS=1 %command%`

Ускорение Расман

<https://aur.archlinux.org/packages/powerpill/> # Установка как ханмод. Спасибо Zee Captain.

В добавляем в `/etc/modprobe.d/nvidia.conf` # Сасибо Вася Стельмачёнок

```
options nvidia NVreg_UsePageAttributeTable=1 NVreg_EnableMSI=1
NVreg_InitializeSystemMemoryAllocations=0 NVreg_EnableStreamMemOPs=1
NVreg_EnablePCleGen3=1
```

NVreg_EnablePCleGen3 (включать только если есть) - Включает поддержку PCIe Gen 3.x. Если система поддерживает эту высокоскоростную шину 8GT, тогда включите ее с помощью этого параметра. Если этот параметр включен, но система не поддерживает Gen 3.0, тогда система может повести себя странно и нестабильно. Некоторые пользователи даже сообщали о повреждении устройства вследствие включения этого параметра при отсутствующей надлежащей поддержке. По умолчанию драйвер Nvidia использует PCIe Gen 2.x из соображений совместимости.

NVreg_UsePageAttributeTable - Это один из последних и новейших параметров драйвера Nvidia. Он позволяет драйверу извлечь максимум из технологии PAT — нового метода выделения памяти, заменяющего более старый метод Memory Type Range Register (MTRR). Метод PAT создает таблицу типа раздела, расположенную по определенному адресу, указанному в регистре, и использует архитектуру памяти и набор инструкций быстрее и

эффективнее. Этот параметр можно включить, если компьютер поддерживает PAT и эта возможность включена в ядре. Включение этого параметра при отсутствующей поддержке PAT может привести к нестабильной работе и даже сбоям системы, поэтому будьте осторожны. (По умолчанию - 0)

NVreg_InitializeSystemMemoryAllocations - Заставляет драйвер NVIDIA очищать системную память перед использованием ее для GPU. Выключение этого параметра может несколько повысить производительность, но ценой повышенного риска с точки зрения безопасности. По умолчанию драйвер очищает выделенную память, обнуляя ее содержимое