# Ручная сборка Arch ISO (Headless + WebUI)

## Этап 1. Подготовка диска sdb на хосте tom_1

Выполняется на живом хосте tom_1 (IP: 192.168.1.72). Диск sdb выступает временным накопителем-донором. с доступом в интернет.

### 1.1 Создание рабочих директорий

#bash

```
mkdir -p ~/original_iso_image/ ~/new_iso/
```

### 1.2 Разметка виртуального диска sdb (будущий arch-flash)

скрипт создаст правильную структуру для UEFI + BIOS:

sdb1 — BIOS Boot (1 МБ) → для загрузки старого BIOS.
sdb2 — EFI System (1 ГБ, FAT32) → для загрузки UEFI (ваш boot).
sdb3 — Linux root (Все оставшееся место, BTRFS)

Используйте один из следующих вариантов размеи дисков с использованием утилит fdisk или sfdisk

1.2.1 Использование утилиты fdisk

#bash

```
sudo fdisk /dev/sdb <<EOF
g
n
1

+1M
t
4
n
2

+1G
t
2
```

```
1
n
3



w
EOF
```

1.2.2 Использование утилиты sfdisk

#bash

```
sudo sfdisk /dev/sdb << EOF
label: gpt
device: /dev/sdb
unit: sectors

/dev/sdb1 : start=        2048, size=        2048,
type=21686148-6449-6E6F-744E-656564454649
/dev/sdb2 : start=        4096, size=     2097152, type=C12A7328-
F81F-11D2-BA4B-00A0C93EC93B
/dev/sdb3 : start=     2101248, size=           +,
type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
EOF
```

## 1.3 Форматирование разделов с метками (LABEL)

1.3.1 Назначение метки ARCH_BOOT для UEFI-загрузчикаFAT32:

#bash

```
sudo mkfs.vfat -F 32 -n "ARCH_BOOT" /dev/sdb2
```

1.3.2 Назначение метки ARCH_ROOT для корневой системы BTRFS:

#bash

```
sudo mkfs.btrfs -f -L "ARCH_ROOT" /dev/sdb3
```

*(Утилита isohybrid из пакета syslinux сама запишет туда MBR-загрузчик на этапе сборки ISO).*

1.3.4 Проверка текущего монтированияЧтобы загрузчик установился корректно, разделы должны быть смонтированы внутри вашей среды arch-chroot. Обычная структура выглядит так:

/mnt — ваш корень BTRFS (sdb3) /mnt/boot — ваш boot-раздел FAT32 (sdb2)

Выполните команду для проверки текущего дерева монтирования:

#bash

```
lsblk -f
```

Что искать в выводе: Убедитесь, что у /dev/sdb3 в колонке MOUNTPOINTS указан / (или /mnt, если вы ещё снаружи chroot). Убедитесь, что у /dev/sdb2 указан /boot (или /mnt/boot). Раздел sdb1 (bios boot) должен оставаться пустым и не смонтированным.

1.3.6 Автоматическая генерация fstab

Важно: Эту команду нужно выполнять снаружи chroot-окружения (внутри установочной флешки), когда все ваши разделы уже смонтированы в папку /mnt. Выполните команду:

#bash

```
genfstab -U /mnt >> /mnt/etc/fstab
```

*(Флаг -U указывает утилите использовать уникальные UUID разделов вместо имен вроде /dev/sdb3. Это гарантирует, что система загрузится, даже если вы вставите диск в другой ПК или другой SATA-разъем.)*

Проверка результата Зайдите внутрь установленной системы (если выходили):

#bash

```
arch-chroot /mnt
```

И откройте файл для проверки:

#bash

```
cat /etc/fstab
```

Если вы ставили систему без подтомов (прямо в корень BTRFS), ваш файл должен выглядеть примерно так:text

#bash

```
# /dev/sdb3 (Root-раздел BTRFS)
UUID=1234abcd-5678-efgh-ijkl-90abcdef1234   /           btrfs
rw,relatime,space_cache=v2,subvolid=5   0 0
```

```
# /dev/sdb2 (Boot-раздел FAT32)
UUID=A1B2-C3D4                                /boot       vfat
rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shortnam
e=mixed,utf8,errors=remount-ro   0 2
```

⚠ Оптимизация для BTRFS (Рекомендуется)
Стандартные настройки genfstab рабочие, но для BTRFS (особенно если у вас SSD-диск) крайне полезно вручную отредактировать опции монтирования для корня (/).
Откройте файл через nano /etc/fstab и добавьте в строку с btrfs следующие параметры через запятую: ssd — включает оптимизацию под твердотельные накопители (если у вас SSD).
compress=zstd — включает прозрачное сжатие данных (сильно экономит место и продлевает жизнь SSD).

Пример оптимизированной строки:

#bash

```
UUID=...   /   btrfs   rw,noatime,compress=zstd,ssd,space_cache=v2
0
```

# Этап 2. Сборка RootFS внутри нового образа (~/new_iso/)

Формируем структуру будущего ISO-образа без использования утилиты archiso.

## 2.1 Монтирование нового диска для заливки системы

#bash

```
mkdir -p ~/new_iso
sudo mount /dev/sdb3 ~/new_iso        # Корень BTRFS монтируем в корень
сборки
sudo mkdir -p ~/new_iso/boot
sudo mount /dev/sdb2 ~/new_iso/boot  # FAT32 монтируем в boot
```

## 2.2 копируем корень tom_1, исключая виртуальные и временные папки.

#bash

```
sudo rsync -aAXv --
exclude={"/dev/*","/proc/*","/sys/*","/tmp/*","/run/*","/mnt/*","/media
/*","/lost+found","/home/*/.cache/*","~/new_iso/*","~/original_iso_imag
e/*"} / ~/new_iso/
```

# Этап 3. Настройка системы через Chroot (chroot ~/new_iso/)

Переходим внутрь создаваемой системы для изоляции настроек.

## 3.1 Вход в окружение

#bash

```
sudo arch-chroot ~/new_iso/
```

## 3.2 Время и Офлайн-режим

3.2.1 Настройка времени (Универсальное UTC + запрет синхронизации):

#bash

```
ln -sf /usr/share/zoneinfo/UTC /etc/localtime
hwclock --systohc
systemctl disable systemd-timesyncd
systemctl mask systemd-timesyncd
```

## 3.3 Настройка статического офлайн-репозитория

3.3.1 Создаем локальный репозиторий внутри образа из кэша tom_1, База пакетов уже скопирована вместе с /var/cache/pacman/pkg/. Создаем локальную БД:

#bash

```
cd /var/cache/pacman/pkg/
repo-add custom.db.tar.gz *.pkg.tar.zst
```

3.3.2 Настройка секции [custom] в pacman.conf для полной изоляции Чтобы система на tom_2 гарантированно не ломилась в сеть и брала пакеты только из локального кэша, мы полностью

отключаем внешние репозитории и зеркала.

3.3.3 Настраиваем pacman.conf внутри образа, чтобы он смотрел только в локальный кэш:
3.3.3.1 Откройте конфигурационный файл:

#bash

```
sudo nano /etc/pacman.conf
```

3.3.4 Настройка файла:

3.3.4.1 Найди и закомментируй (поставь # в начале строки) все стандартные репозитории: [core], [extra], [community]. Вместе с ними закомментируй строки Include = /etc/pacman.d/mirrorlist.

3.3.4.2 В самый конец файла добавь твою локальную секцию [custom]. Итоговый блок репозиториев в /etc/pacman.conf должен выглядеть строго так:

pacman.conf

```
# Полностью отключаем внешние репозитории
#[core]
#Include = /etc/pacman.d/mirrorlist

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

# Добавляем изолированный локальный репозиторий
[custom]
SigLevel = Optional TrustAll
Server = file:///var/cache/pacman/pkg
```

3.3.5 Проверка внутри Chroot ==== После сохранения файла обязательно обнови локальную базу данных, чтобы проверить работу:

#bash

```
pacman -Sy
```

*(Результат: Система должна моментально считать базу данных custom.db напрямую из папки без единого сетевого запроса.)*

## 3.4 Настройка сети (Статический IP флешки в ОЗУ)

Прописываем параметры для работы в ОЗУ (IP: 192.168.1.150). Создаем профиль systemd-networkd:

#bash

```
rm -f /etc/systemd/network/*
nano /etc/systemd/network/20-wired.network
```

Содержимое:

20-wired.network

```
[Match]
Name=en*
Name=eth*

[Network]
Address=192.168.1.150/24
Gateway=192.168.1.1
DNS=1.1.1.1
```

## 3.5 Включение сервисов (Nginx порт 7000 + SSH)

3.5.1 Правим порт Nginx:

#bash

```
nano /etc/nginx/nginx.conf
```

nginx.conf

```
listen       7000;
server_name  localhost;
```

3.5.2 Активируем автозапуск:

#bash

```
systemctl enable systemd-networkd systemd-resolved sshd nginx
```

## 3.6 Настройка Swap в ОЗУ (zram-generator)

#bash

```
nano /etc/systemd/zram-generator.conf
```

Содержимое:

zram-generator.conf

```
[zram0]
zram-size = ram / 2
compression-algorithm = zstd
```

## 3.7 Перезапись fstab под новые метки (LABEL)

#bash

```
# Так как rsync скопировал реальный fstab с UUID хоста tom_1,
# мы полностью очищаем его внутри chroot, чтобы Live-система на tom_2
работала в ОЗУ без привязки к дискам.
true > /etc/fstab
```

( Если система при загрузке в ОЗУ затребует UUID дисков, то пробуем этот пункт 3.7 Перезапись fstab под новые метки (LABEL):

#bash

```
cat <<EOF > /etc/fstab
LABEL=ARCH_ROOT    /        btrfs    defaults,noatime,compress=zstd
0 0
LABEL=ARCH_BOOT    /boot    vfat     defaults,noatime
0 2
EOF
```

)

## 3.8 Установка и обновление EFI-загрузчика

#bash

```
bootctl install
```

Выход из chroot:

#bash

```
exit
```

# Этап 4. Конфигурация загрузчика systemd-boot и Syslinux

Установка загрузчика (Выполняется на хосте tom_1. На этом этапе разделы диска sdb все еще примонтированы в ~/new_iso/!):

## 4.1 Настройка параметров загрузки (loader.conf)

#bash

```
sudo nano ~/new_iso/boot/loader/loader.conf
```

Содержимое (выбор 3 секунды):

loader.conf

```
inidefault arch
timeout 3
console-mode max
editor no
```

## 4.2 Настройка записи загрузки параметры ядра (arch.conf)

Привязка к глобальной метке ARCH_ROOT, COM-порт и отключение прерываний Hyper-V:

#bash

```
sudo nano ~/new_iso/boot/loader/entries/arch.conf
```

Содержимое:

ini

```
# Для ~/new_iso/boot/loader/entries/arch.conf и isolinux.cfg
title Arch Linux Custom Live (SquashFS Manual)
linux /vmlinuz-linux
initrd /initramfs-linux.img
options archisolabel==ARCH_202605
archisobasedir=/arch/x86_64/airootfs.sfs rw console=ttyS0,115200n8
earlyprintk=ttyS0,115200 hv_utils.disable_gpadl_match=1
```

## 4.3 Подготовка Syslinux (BIOS) для физического железа

#bash

```
sudo pacman -S syslinux --noconfirm
sudo mkdir -p ~/new_iso/boot/syslinux
sudo cp
/usr/lib/syslinux/bios/{isolinux.bin,ldlinux.c32,libcom.c32,libutil.c32
,vesamenu.c32} ~/new_iso/boot/syslinux/
```

#bash

```
sudo nano ~/new_iso/boot/syslinux/isolinux.cfg
```

Содержимое:

ini

```
UI vesamenu.c32
PROMPT 0
TIMEOUT 30
DEFAULT arch

LABEL arch
LINUX /vmlinuz-linux
INITRD /initramfs-linux.img
APPEND archisolabel==ARCH_202605
archisobasedir=/arch/x86_64/airootfs.sfs rw console=ttyS0,115200n8
earlyprintk=ttyS0,115200 hv_utils.disable_gpadl_match=1
```

## 4.4. Настройка родного initramfs для поддержки SquashFS и OverlayFS

Чтобы система загрузилась в режиме «только чтение» из SquashFS, но позволяла изменять файлы в ОЗУ (принцип LiveCD), стандартный initramfs должен уметь работать с модулями loop и overlay.

Выполните следующие действия на хосте tom_1: Зайдите в окружение Chroot:

#bash

```
sudo arch-chroot ~/new_iso/
```

Включите поддержку модулей в конфигурации сборщика:

#bash

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

Найдите строку MODULES=(...) и добавьте туда модули для работы с петлевыми устройствами и сжатыми файловыми системами:

#bash

```
MODULES=(loop overlay squashfs vfat btrfs)
```

Проверьте хуки: В строке HOOKS=(...) убедитесь, что присутствуют systemd и block:

#bash

```
HOOKS=(base udev modconf memdisk archiso_loop_mnt archiso
archiso_pxe_common archiso_pxe_nbd archiso_pxe_http archiso_pxe_nfs
block filesystems keyboard)
```

Пересоберите initramfs: Генерируем новый образ загрузки, который теперь аппаратно готов смонтировать наш будущий .sfs файл:

#bash

```
mkinitcpio -p linux
```

Выйдите из chroot:

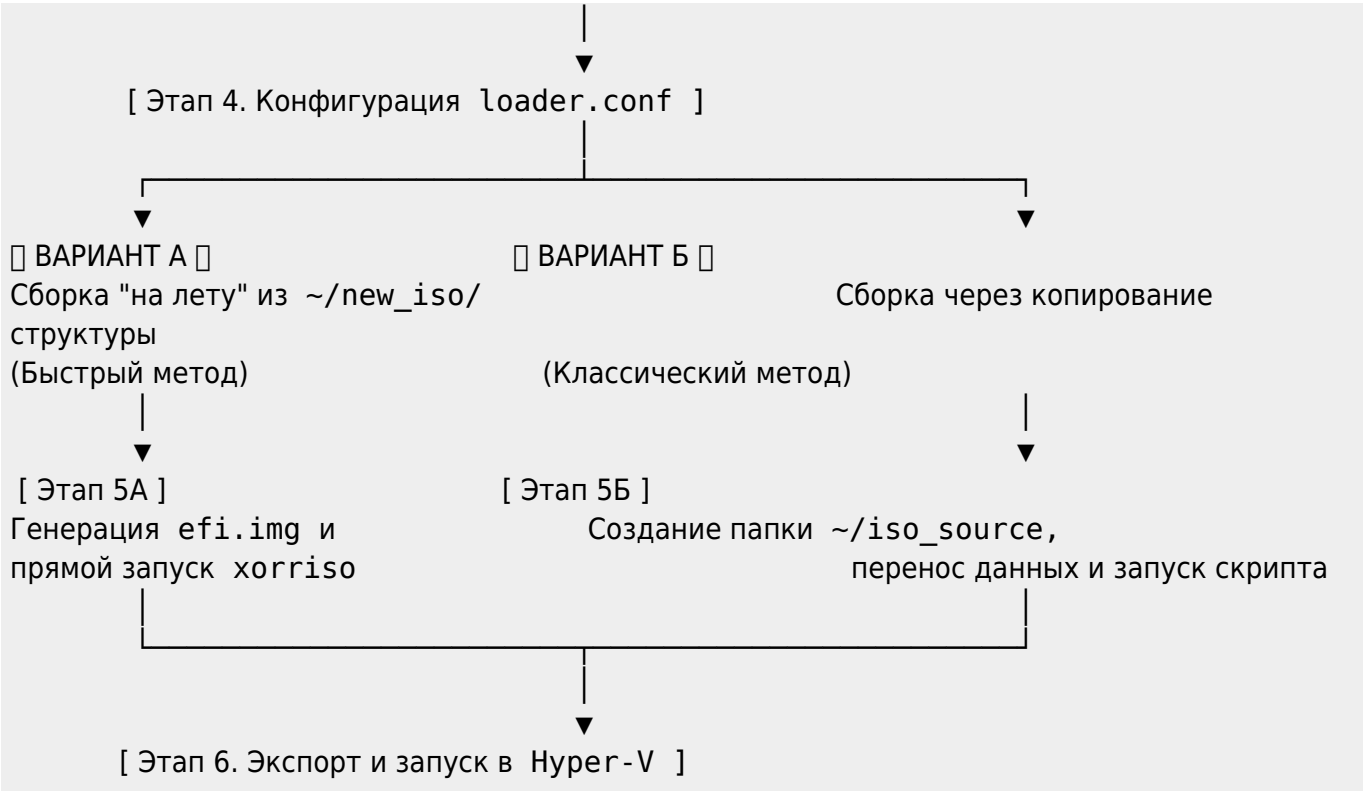
#bash

```
exit
```

# Этап 5. Выбор варианта Генерация UEFI-образа, SquashFS и сборка ISO для через xorriso

 ВАРИАНТ А 
Сборка «на лету» прямо из ~/new_iso/ (Быстрый метод)
Метод собирает ISO прямо из рабочей директории, где смонтирован диск sdb.
*(Идеально для быстрых тестов.)*
 ВАРИАНТ Б 
Сборка через изолированную структуру ~/iso_source (Классический метод)

```
[ Этап 3. Настройка через Chroot ]
```




# Этап 5A. Генерация UEFI-образа, SquashFS и сборка ISO для через xorriso

 ВАРИАНТ А 
Сборка «на лету» прямо из ~/new_iso/ (Быстрый метод)
Метод собирает ISO прямо из рабочей директории, где смонтирован диск sdb.
*(Идеально для быстрых тестов.)*

## 5A.1 Создание внутренней структуры Arch ISO и сжатие RootFS

# 1. Создаем целевой каталог внутри сборки

#bash

```
mkdir -p ~/new_iso/arch/x86_64/
```

# 2. Упаковываем RootFS во временный файл в /tmp/ хоста

#bash

```
sudo mksquashfs ~/new_iso /tmp/airootfs.sfs -comp zstd -noappend -e
arch tmp lost+found
```

# 3. Перемещаем готовый изолированный образ внутрь структуры

#bash

```
sudo mv /tmp/airootfs.sfs ~/new_iso/arch/x86_64/airootfs.sfs
```

## 5A.2 Создание внутреннего efi.img внутри структуры

# 1. Создаем образ efi.img во временной папке хоста в ~/new_iso/boot

#bash

```
sudo dd if=/dev/zero of=~/new_iso/boot/efi.img bs=1M count=64
status=none
sudo mkfs.vfat -F 16 -n "ARCH_202605" ~/new_iso/boot/efi.img >
/dev/null
```

# 2. Монтируем его для наполнения

#bash

```
mkdir -p /tmp/efi_mnt
sudo mount -o loop /tmp/efi.img /tmp/efi_mnt
```

# 3. Копируем файлы загрузчика ИЗ смонтированного диска во временный efi.img

#bash

```
sudo mkdir -p /tmp/efi_mnt/EFI/BOOT /tmp/efi_mnt/loader/entries
sudo cp ~/new_iso/boot/EFI/BOOT/BOOTX64.EFI /tmp/efi_mnt/EFI/BOOT/
sudo cp ~/new_iso/boot/loader/loader.conf /tmp/efi_mnt/loader/
sudo cp ~/new_iso/boot/loader/entries/arch.conf
/tmp/efi_mnt/loader/entries/
```

# 4. Размонтируем и удаляем точку монтирования

#bash

```
sudo umount /tmp/efi_mnt
rmdir /tmp/efi_mnt
```

## 5A.3 Прямой запуск xorriso сборки гибридного ISO

# Запускаем сборку из директории ~/new_iso/. Результат положим в корень домашней

директории.

#bash

```
sudo xorriso -as mkisofs \
  -iso-level 3 \
  -full-iso9660-filenames \
  -volid "ARCH_202605" \
  -eltorito-boot boot/syslinux/isolinux.bin \
  -eltorito-catalog boot/syslinux/boot.cat \
  -no-emul-boot -boot-load-size 4 -boot-info-table \
  -isohybrid-mbr /usr/lib/syslinux/bios/isohdpfx.bin \
  -eltorito-alt-boot \
  -e boot/efi.img \
  -no-emul-boot -isohybrid-gpt-basdat \
  -hide EFI \
  -hide loader \
  -output ~/ARCH_202605.iso \
  ~/new_iso/
```

### 5А.4 Безопасное размонтирование диска донора

# Только теперь, когда ISO-образ успешно создан, освобождаем накопитель:

#bash

```
# Размонтируем разделы диска sdb, которые были подключены к хосту tom_1
sudo umount ~/new_iso/boot
sudo umount ~/new_iso
rm -f /tmp/efi.img
```

> Примечание: «Этап сборки завершен. Пропустите Вариант Б и переходите сразу к Этапу 6».

## Этап 5Б. Подготовка структуры и сборка ISO через xorriso

 ВАРИАНТ Б Сборка через изолированную структуру ~/iso_source (Классический метод)

> Этот этап применяется, если не использовался Вариант А, позволяя упаковать систему в SquashFS и создать гибридный ISO-образ, используя временную директорию ~/iso_source.

*(Метод копирует все данные в отдельную директорию на хосте, освобождая диск донор sdb сразу.)*

### 5Б.1 Создание структуры и копирование данных

Создаем рабочую директорию и синхронизируем в нее данные, исключая временный образ EFI:
# Создаем чистую директорию

#bash

```
mkdir -p ~/iso_source
```

# Синхронизируем содержимое sdb в папку сборки для xorriso

#bash

```
sudo rsync -aAXv --exclude={"/boot/efi.img"} ~/new_iso/ ~/iso_source/
```

### 5Б.2 Освобождение диска-донора

Размонтируем исходный диск sdb, так как данные уже скопированы в iso_source:

#bash

```
sudo umount ~/new_iso/boot
sudo umount ~/new_iso
```

### 5Б.3 Упаковка RootFS в SquashFS

Упаковываем файловую систему в сжатый образ airootfs.sfs с использованием zstd, исключая служебные каталоги:

# 1. Создаем целевой каталог внутри изолированной структуры

#bash

```
mkdir -p ~/iso_source/arch/x86_64/
```

# 2. Упаковываем RootFS во временный файл из папки ~/iso_source, исключая arch, tmp и lost+found

#bash

```
sudo mksquashfs ~/iso_source /tmp/airootfs.sfs -comp zstd -noappend -e
arch tmp lost+found
```

# 3. Перемещаем готовый изолированный образ внутрь правильной структуры

#bash

```
sudo mv /tmp/airootfs.sfs ~/iso_source/arch/x86_64/airootfs.sfs
```

## 5Б.4 Создание скрипта сборки ISO глобальной меткой ARCH_202605.

Так как мы собираем чистый гибридный UEFI/BIOS образ вручную (без archiso), нам нужен готовый скрипт, который соберет структуру папки ~/iso_source/ в правильный .iso файл. Выполняется на хосте tom_1, вне Chroot.

### 5Б.4.1. Создай файл скрипта

#bash

```
nano ~/build_iso.sh
```

### 5Б.4.2. Вставь в него следующий код

#bash

```
#!/bin/bash

# Настройка путей
SOURCE_DIR="$HOME/iso_source"
OUTPUT_ISO="$HOME/ARCH_202605.iso"
VOLUME_ID="ARCH_202605"
TMP_EFI_MNT="/tmp/efi_mnt"

echo "=== Старт автоматической сборки гибридного ISO (BIOS + UEFI) ==="

# 1. Проверка утилит
if ! command -v xorriso &> /dev/null; then
    echo "Ошибка: xorriso не установлен. Выполните: sudo pacman -S
xorriso"
    exit 1
fi
```

```
# 2. Безопасный бэкап файлов загрузчика перед операциями, если sdb2 еще примонтирован
# Если sdb2 уже размонтирован, скрипт возьмет файлы из структуры копии
echo "→ Синхронизация файлов загрузчика..."
mkdir -p /tmp/loader_backup/EFI/BOOT
mkdir -p /tmp/loader_backup/loader/entries

if mountpoint -q ~/new_iso/boot; then
    cp -r ~/new_iso/boot/EFI /tmp/loader_backup/
    cp -r ~/new_iso/boot/loader /tmp/loader_backup/
else
    if [ -d "$SOURCE_DIR/boot/EFI" ]; then
        cp -r "$SOURCE_DIR/boot/EFI" /tmp/loader_backup/
        cp -r "$SOURCE_DIR/boot/loader" /tmp/loader_backup/
    else
        echo "Критическая ошибка: Файлы UEFI-загрузчика не найдены ни в ~/new_iso/boot, ни в $SOURCE_DIR/boot!"
        exit 1
    fi
fi

# 3. Генерация EFI-образа для загрузки UEFI
echo "→ Подготовка EFI boot image..."
rm -f "$SOURCE_DIR/boot/efi.img"
dd if=/dev/zero of="$SOURCE_DIR/boot/efi.img" bs=1M count=64 status=none
mkfs.vfat -F 16 -n "ARCH_202605" "$SOURCE_DIR/boot/efi.img" > /dev/null

# 4. Монтируем efi.img и копируем туда файлы загрузчика systemd-boot
mkdir -p /tmp/efi_mnt
sudo mount -o loop "$SOURCE_DIR/boot/efi.img" $TMP_EFI_MNT


# 5. Проверяем успешность монтирования перед тем, как работать с директорией
if mountpoint -q $TMP_EFI_MNT; then
    echo "→ Наполнение efi.img файлами загрузчика..."
    sudo mkdir -p $TMP_EFI_MNT/EFI/BOOT
    sudo mkdir -p $TMP_EFI_MNT/loader/entries

    # Копируем из гарантированного бэкапа
    sudo cp /tmp/loader_backup/EFI/BOOT/BOOTX64.EFI $TMP_EFI_MNT/EFI/BOOT/
    sudo cp /tmp/loader_backup/loader/loader.conf $TMP_EFI_MNT/loader/
    sudo cp /tmp/loader_backup/loader/entries/arch.conf $TMP_EFI_MNT/loader/entries/

    sudo umount $TMP_EFI_MNT
    rmdir $TMP_EFI_MNT
    rm -rf /tmp/loader_backup
else
    echo "Критическая ошибка: Не удалось примонтировать efi.img!"
```

```
    rmdir $TMP_EFI_MNT
    exit 1
fi

# 6. Безопасная проверка готовности каталога boot перед сборкой ISO
# Вместо деструктивного удаления файлов, мы просто проверяем наличие efi.img
echo "→ Проверка загрузочной структуры в boot..."
if [ ! -f "$SOURCE_DIR/boot/efi.img" ]; then
    echo "Критическая ошибка: efi.img отсутствует в $SOURCE_DIR/boot/!"
    exit 1
fi

# Чтобы xorriso не затягивал дублирующие папки EFI и loader в корень ISO
# (они уже упакованы внутрь efi.img), мы укажем утилите xorriso исключить их
# прямо во время сборки на Шаге 7 с помощью флага -hide.

# 7. Сборка полноценного гибридного ISO через xorriso
echo "→ Запуск xorriso (Сборка гибридного образа)..."
xorriso -as mkisofs \
  -iso-level 3 \
  -full-iso9660-filenames \
  -volid "$VOLUME_ID" \
  -eltorito-boot boot/syslinux/isolinux.bin \
  -eltorito-catalog boot/syslinux/boot.cat \
  -no-emul-boot -boot-load-size 4 -boot-info-table \
  -isohybrid-mbr /usr/lib/syslinux/bios/isohdpfx.bin \
  -eltorito-alt-boot \
  -e boot/efi.img \
  -no-emul-boot -isohybrid-gpt-basdat \
  -hide EFI \
  -hide loader \
  -output "$OUTPUT_ISO" \
  "$SOURCE_DIR/"


if [ $? -eq 0 ]; then
    echo "=== Сборка успешно завершена! ==="
    echo "Файл образа: $OUTPUT_ISO"
    echo "Этот образ готов к записи через Rufus (в режиме DD/ISO) для флешек
ИЛИ прямого монтирования в Hyper-V Gen1/Gen2."
else
    echo "=== Ошибка при сборке ISO ==="
    exit 1
fi
```

### 5Б.5 Запуск сборки

Делаем скрипт исполняемым и запускаем его для создания финального образа:

#bash

```
chmod +x ~/build_iso.sh
~/build_iso.sh
```

*(Примечание: Параметры xorriso могут адаптироваться под структуру папки boot вашего диска sdb).*

Скрипт автоматически упакует систему, создаст правильный UEFI-загрузочный сектор efi.img с твоей глобальной меткой ARCH_202605 и положит готовый файл в твою домашнюю директорию.

# Этап 6. Экспорт ISO в Windows и монтирование в CD-привод Hyper-V (tom_2)

## 6.1 Передача файла на Windows-хост

Используте WinSCP на Windows:

cmd

```
pscp eva@192.168.1.72:/home/eva/ARCH_202605.iso C:\ISO\
```

## 6.2 Запись на arch-flash (опционально, для физ. теста)

Открываешь Rufus, выбираешь флешку, выбираешь созданный ARCH_202605.iso и пишешь в режиме DD/ISO.

## 6.3 Монтирование нового ISO-образа в CD-привод Hyper-V (для tom_2)

Открой Диспетчер Hyper-V. Выбери изолированную виртуальную машину tom_2. Нажми Параметры (Settings) → vIDE-контроллер или SCSI-контроллер (в зависимости от поколения ВМ Gen1/Gen2). Выбери Накопитель DVD (DVD Drive). Установи переключатель в положение Файл образа (ISO) (Image file). Нажми Обзор (Browse) и укажи путь к скачанному файлу C:\ISO\ARCH_202605.iso. Нажми Применить (Apply) и запусти ВМ tom_2.

From:
https://wwoss.direct.quickconnect.to/ - **worldwide open-source software**

Permanent link:
**https://wwoss.direct.quickconnect.to/doku.php?id=30.05.2026&rev=1780179136**

Last update: **2026/05/31 01:12**