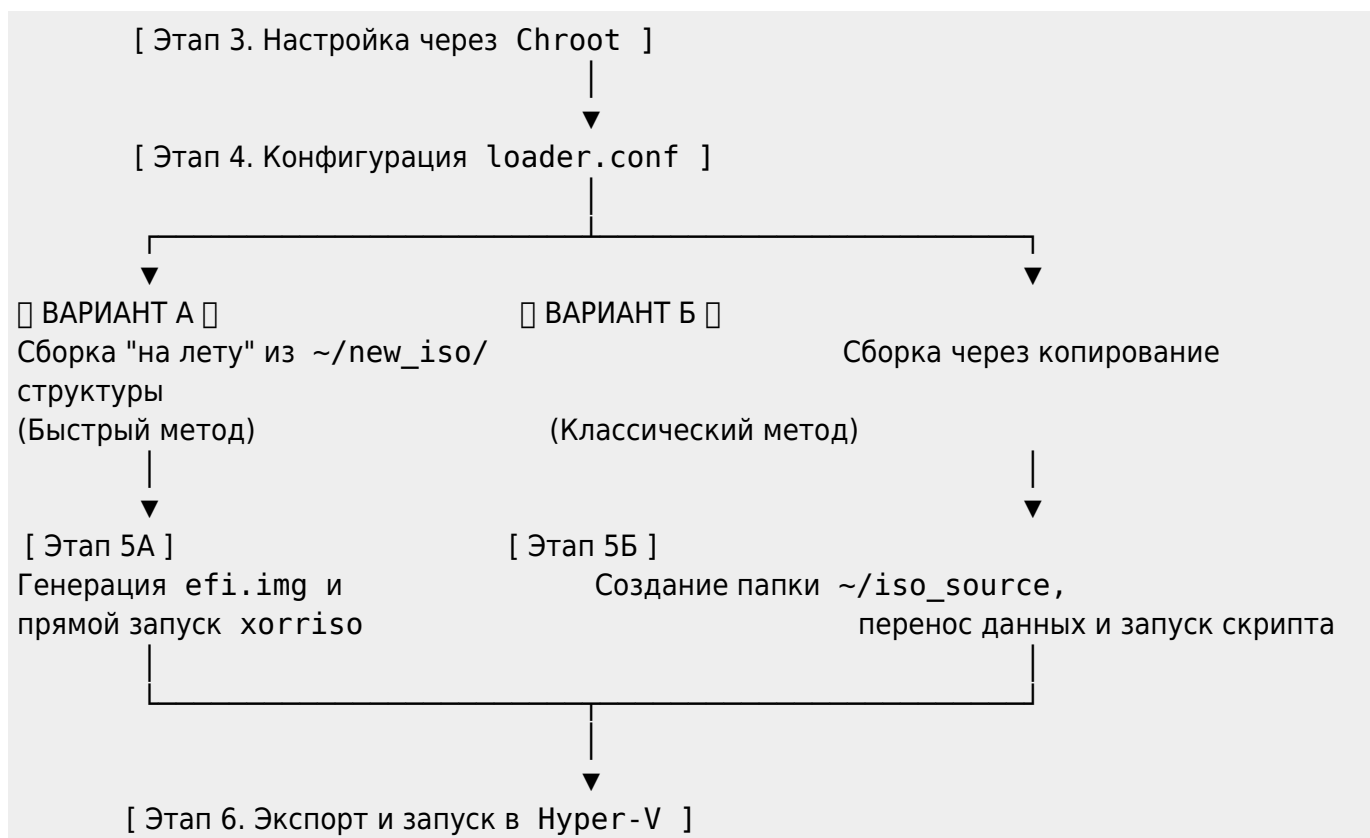


1.3.1. Распределение ролей при работе в Hyper-V (Virtual Machine) tom_1 - эталонный хост, основная тестовая VM в Hyper-V для сборки iso образа, с доступом в интернет, с рабочим arch linux, на котором разверачивается Nginx (порт 7000) и работает SSH для PuTTY, созданы пользователи root и eva и им заданы пороли. arch-flash - виртуальный диск в Hyper-V (носитель с записанным iso образом), куда мы через rufus записываем, созданный нами ISO-образ на tom_1 tom_2 - изолированный виртуальный хост, VM в Hyper-V без доступа в интернет, куда мы подключим arch-flash и развернем arch linux, на котором развернут Nginx (порт 7000) и работает SSH для PuTTY, созданы пользователи root и eva и им заданы пороли. Добавление ролей при работе с физическим железом usb-arch-server - физический usb носитель arch-server - физический сервер (supermicro/hp/no-name)

1.3.2. Администраторы сервера root - по умолчанию в Arch linux eva - из установленного Arch linux admin - новый создаваемый администратор ~/original_iso_image/ - сюда грузим из интернета оригинальный образ ~/new_iso/ - тут собираем новый



----- 1. Дублирование сборки ISO (Этап 5А против Этапа 5): мы включаем 2 вида: либо Этап 5А / либо Этапа 5

2. Смешение сущностей «RootFS» и «Готовый ISO»: ~/new_iso/ в ней мы собираем новый образ sdb1 - это boot для arch-flash sdb2 - это root для arch-flash

3. Ошибки в синтаксисе путей: В пункте 3.3.4.2 в параметре Server указано: В пункте 3.3.4.2 в параметре Server указано через notepad++ указано верно Server = file:/var/cache/pacman/pkg. - видимо на сайте вики отображает через *

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: `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` `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` `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-загрузчика FAT32: `sudo mkfs.vfat -F 32 -n «ARCH_BOOT» /dev/sdb2` 1.3.2 Назначение метки `ARCH_ROOT` для корневой системы BTRFS: `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 монтирования Чтобы 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: `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: `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): `arch-chroot /mnt` И откройте файл для проверки: `cat /etc/fstab` Если вы ставили систему без подтомов (прямо в корень BTRFS), ваш файл должен выглядеть примерно так: `text # /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,icharset=ascii,shortname=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: `UUID=... / btrfs rw,noatime,compress=zstd,ssd,space_cache=v2 0 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: `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. `sudo rsync -aXv --exclude={«/dev/*»,«/proc/*»,«/sys/*»,«/tmp/*»,«/run/*»,«/mnt/*»,«/media/*»,«/lost+found»,«/home/*/.cache/*»,«~/new_iso/*»,«~/original_iso_image/*»} / ~/new_iso/` ===== Этап 3. Настройка системы через `Chroot` (`chroot ~/new_iso/`) Переходим внутрь создаваемой системы для изоляции настроек. 3.1 Вход в окружение: `sudo arch-chroot ~/new_iso/` 3.2 Время и Офлайн-режим 3.2.1 Настройка времени (Универсальное UTC + запрет синхронизации): `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: `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 ломалась 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: `sudo nano /etc/pacman.conf`

3.3.4 Настройка файла:

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

3.3.4.2 В самый конец файла добавьте вашу локальную секцию `[custom]`. Итоговый блок репозитория в `/etc/pacman.conf` должен выглядеть строго так: `#ini # 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 обновите локальную базу данных, чтобы проверить работу: `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`: `rm -f /etc/systemd/network/* nano /etc/systemd/network/20-wired.network`

Содержимое: `[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: `nano /etc/nginx/nginx.conf`

`listen 7000; server_name localhost;`

3.5.2 Активируем автозапуск: `systemctl enable systemd-networkd systemd-resolved sshd nginx`

3.6 Настройка Swap в ОЗУ (`zram-generator`): `nano /etc/systemd/zram-generator.conf`

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

3.7 Перезапись `fstab` под новые метки (LABEL): `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: `bootctl install`

Выход из chroot: `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`): `sudo nano ~/new_iso/boot/loader/loader.conf`

Содержимое (выбор 3 секунды): `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: `sudo nano ~/new_iso/boot/loader/entries/arch.conf`

Содержимое: `ini title Arch Linux Custom Live (SquashFS Manual) linux /boot/vmlinuz-linux initrd /boot/initramfs-linux.img options img_label=ARCH_202605 img_loop=/arch/airootfs.sfs rw console=ttyS0,115200n8 earlyprintk=ttyS0,115200 hv_utils.disable_gpadl_match=1`

4.3 Подготовка Syslinux (BIOS) для физического железа: `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/`

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

Содержимое: `ini UI vesamenu.c32 PROMPT 0 TIMEOUT 30 DEFAULT arch`

`LABEL arch`

```
LINUX /boot/vmlinuz-linux
INITRD /boot/initramfs-linux.img
APPEND img_label=ARCH_202605 img_loop=/arch/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: `sudo arch-chroot ~/new_iso/`

Включите поддержку модулей в конфигурации сборщика: `nano /etc/mkinitcpio.conf`

Найдите строку `MODULES=(...)` и добавьте туда модули для работы с петлевыми устройствами и сжатыми файловыми системами: `MODULES=(loop overlay squashfs vfat btrfs)`

Проверьте хуки: В строке `HOOKS=(...)` убедитесь, что присутствуют `systemd` и `block`: `HOOKS=(base systemd autodetect modconf block filesystems keyboard)`

Пересоберите initramfs: Генерируем новый образ загрузки, который теперь аппаратно готов смонтировать наш будущий .sfs файл: `mkinitcpio -p linux`

Выйдите из chroot: `exit`

— ВЫБЕРИТЕ ОДИН ИЗ ВАРИАНТОВ СБОРКИ — □ 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.

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

5А.1 Создание внутренней структуры Arch ISO и сжатие RootFS: # 1. Создаем целевой каталог внутри сборки `mkdir -p ~/new_iso/arch/x86_64/`

```
# 2. Упаковываем RootFS во временный файл в /tmp/ хоста sudo mksquashfs ~/new_iso  
/tmp/airootfs.sfs -comp zstd -noappend -e boot arch tmp lost+found
```

```
# 3. Перемещаем готовый изолированный образ внутрь структуры 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 /tmp (а не в ~/new_iso/boot!) sudo dd if=/dev/zero of=/tmp/efi.img bs=1M count=64 status=none sudo mkfs.vfat -F 16 -n «ARCH_202605» /tmp/efi.img > /dev/null

```
# 2. Монтируем его для наполнения 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 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 sudo umount /tmp/efi_mnt rmdir /tmp/efi_mnt
```

5A.3 Прямой запуск xorriso сборки гибридного ISO для Варианта А: # Запускаем сборку прямо из директории ~/new_iso/. Результат положим в корень домашней директории. 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/isohdpx.bin \ -eltorito-alt-boot \ -e /tmp/efi.img \ -no-emul-boot -isohybrid-gpt-basdat \ -output ~/ARCH_202605.iso \ ~/new_iso/

5A.4 Безопасное размонтирование диска донора # Только теперь, когда ISO-образ успешно создан, освобождаем накопитель: # 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».

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

Этап 5Б. Подготовка структуры и сборка ISO через xorriso 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 mkdir -p ~/iso_source

```
# Синхронизируем содержимое sdb в папку сборки для xorriso sudo rsync -aXv  
-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: 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 сборки mkdir -p ~/new_iso/arch/x86_64/

2. Упаковываем RootFS во временный файл в /tmp/ хоста sudo mksquashfs ~/new_iso /tmp/airootfs.sfs -comp zstd -noappend -e boot arch tmp lost+found

3. Перемещаем готовый изолированный образ внутрь структуры sudo mv /tmp/airootfs.sfs ~/new_iso/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: nano ~/build_iso.sh

5Б.4.2. Вставь в него следующий код: #!/bin/bash

```
# Настройка путей # 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. Очистка каталога boot внутри iso_source от лишних файлов перед сборкой ISO #
Оставляем только то, что нужно syslinux (BIOS) и сам efi.img (UEFI) mv
«$SOURCE_DIR/boot/efi.img» /tmp/efi.img.tmp rm -rf «$SOURCE_DIR/boot/EFI»
«$SOURCE_DIR/boot/loader» mkdir -p «$SOURCE_DIR/boot» mv /tmp/efi.img.tmp
«$SOURCE_DIR/boot/efi.img»
```

```
# 7. Сборка полноценного гибридного ISO через xorriso echo «→ Запуск xorriso (Сборка
гибридного образа)...» xorriso -as mkisofs \
```

1. iso-level 3 \
2. full-iso9660-filenames \
3. volid «\$VOLUME_ID» \
4. eltorito-boot boot/syslinux/isolinux.bin \
5. eltorito-catalog boot/syslinux/boot.cat \
6. no-emul-boot -boot-load-size 4 -boot-info-table \
7. isohybrid-mbr /usr/lib/syslinux/bios/isohdpx.bin \
8. eltorito-alt-boot \
9. e boot/efi.img \
10. no-emul-boot -isohybrid-gpt-basdat \
11. 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: `chmod +x ~/build_iso.sh ~/build_iso.sh`

(Примечание: Параметры horriso 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: Используйте 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 VM 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 VM 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=1780166547>

Last update: **2026/05/30 21:42**

