***arch установить пакет syslinux в авторежиме и проверить установку Установить дополнительно Для установки и первичной настройки загрузчика https://wiki.archlinux.org/title/Syslinux_(%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9) syslinux в неинтерактивном (автоматическом) режиме используйте менеджер пакетов и утилиты конфигурации, выполнив команды sudo pacman -S syslinux –noconfirm sudo extlinux –install /boot/syslinux Для полной проверки правильности установки, настройки и готовности к запуску выполните следующие шаги: 1. Проверка файлов конфигурации: убедитесь, что в директории boot созданы необходимые каталоги: ls -la /boot/syslinux/ 2. Проверка модулей: убедитесь, что файлы ядра syslinux скопировались корректно: ls -la /usr/lib/syslinux/bios/ 3. Генерация таблицы: обновите fstab и проверьте конфигурационный файл syslinux.cfg, чтобы загрузчик знал, на каком диске установлена система.** *arch установить пакет rsync в авторежиме и проверить установку

***Для автоматической установки rsync в Arch Linux и проверки его наличия выполните одну команду в терминале: sudo pacman -Syy –noconfirm rsync && rsync –version Разбор команды sudo pacman -Syy — принудительно обновляет базы данных пакетов. –noconfirm — автоматически одобряет все запросы и отключает интерактивный режим. && — запускает следующую команду только при успешной установке. rsync –version — выводит версию утилиты для проверки её работоспособности.** *arch установить пакет tree в авторежиме и проверить установку

sudo pacman -S –noconfirm tree && tree –version

Разбор команды sudo pacman -Syy — принудительно обновляет базы данных пакетов. –noconfirm — автоматически одобряет все запросы и отключает интерактивный режим. && — запускает следующую команду только при успешной установке. tree –version — выводит версию утилиты для проверки её работоспособности.

***Доустанавливаем утилиту для FAT32 dosfstools — содержит утилиты для создания и проверки файловых систем семейства FAT (включая mkfs.vfat, mkfs.msdos). Это критически важно для создания системного EFI-раздела. btrfs-progs — содержит инструменты для управления файловой системой Btrfs (включая mkfs.btrfs, работу с подтомами и снимками).** *arch установить пакет dosfstools в авторежиме и проверить установку

sudo pacman -S –noconfirm dosfstools && mkfs.vfat –version

Разбор команды sudo pacman -Syy — принудительно обновляет базы данных пакетов. –noconfirm — автоматически одобряет все запросы и отключает интерактивный режим. && — запускает следующую команду только при успешной установке. mkfs.vfat –version — вызывает одну из основных утилит пакета (создание файловой системы FAT) для подтверждения успешной установки.

***arch установить пакет btrfs progs в авторежиме и проверить установку sudo pacman -S –noconfirm btrfs-progs && btrfs –version Разбор команды sudo pacman -Syy — принудительно обновляет базы данных пакетов. –noconfirm — автоматически одобряет все запросы и отключает интерактивный режим. && — запускает следующую команду только при успешной установке. btrfs –version — выводит версию утилиты, подтверждая её готовность к работе.** *arch проверить версию bootctl Поскольку bootctl встроен в пакет systemd, его версия всегда совпадает с версией systemd, установленной в вашей системе. bootctl –version pacman -Q systemd

***arch установить пакет arch-install-scripts в авторежиме и проверить установку sudo pacman -S –noconfirm arch-install-scripts && pacstrap -h sudo pacman -S — запускает установку пакета из официальных репозиториев. –noconfirm — отключает интерактивные запросы для работы в автоматическом режиме. && — переходит к следующему шагу только при успешном завершении установки. pacstrap -h — вызывает справку утилиты pacstrap (главного инструмента этого пакета), подтверждая успешную установку.** *arch установить пакет efibootmgr в авторежиме и проверить установку pacman -S grub efibootmgr –noconfirm для роцессоров Intel и AMD, сразу добавить пакет микрокода: pacman -S intel-ucode –noconfirm pacman -S amd-ucode –noconfirm

grub-install –version

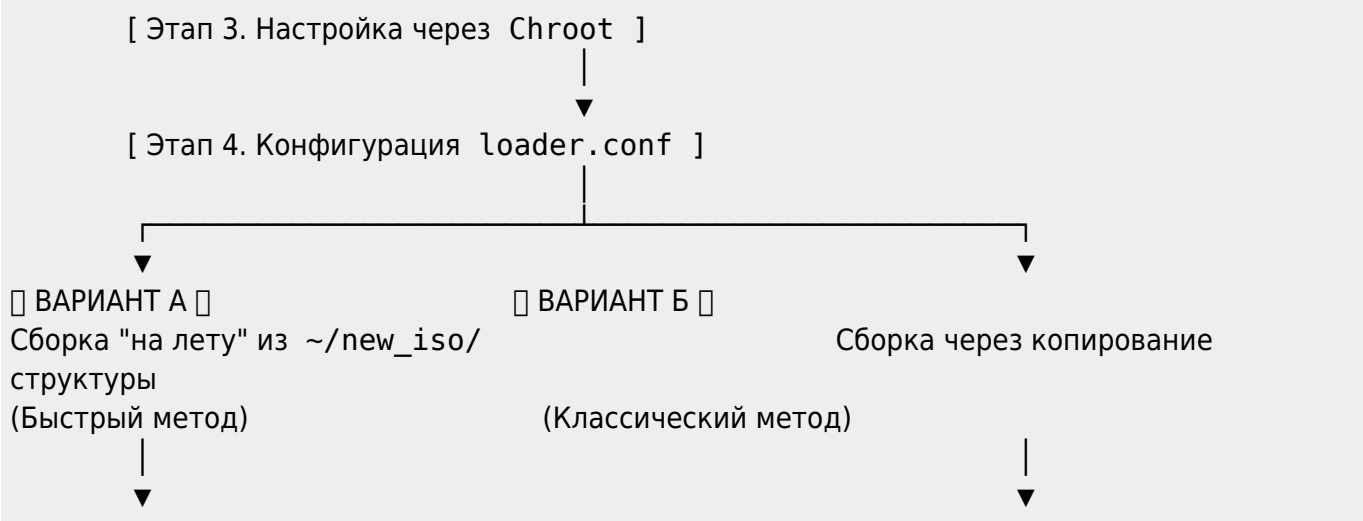
***arch установить пакет os-prober в авторежиме и проверить установку

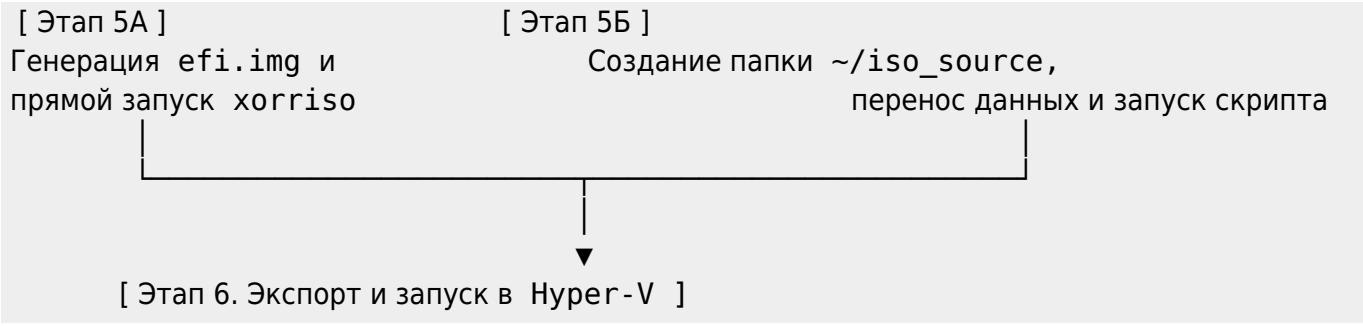
sudo pacman -S os-prober –noconfirm

pacman -Q os-prober

прочти заново https://wwoss.ru/doku.php?id=30.05.2026 и найди ошибки и укажи пути правки как было, как должно быть с указанием пунктов

1.3.1. Распределение ролей при работе в Hyper-V (Virtual Machine) tom_1 - эталонный хост, основная тестовая ВМ 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 ы Hyper-V (носитель с записанным iso образом), куда мы через rufus записываем, созданный нами ISO-образ на tom_1 tom_2 - изолированный виртуальный хост, ВМ 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/ - суда грузим из интернета оригинальный образ ~/new_iso/ - тут собираем новый





————————————————————————————– 1. Дублирование сборки ISO (Этап 4А против Этапа 5): мы влючаем 2 вида: либо Этап 4А / 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++ указанно 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) sudo fdisk /dev/sdb «EOF g n 1 +1M t bios n 2 +1G t 2 uefi n 3 w EOF 1.3 Форматирование разделов с метками (LABEL): 1.3.1 Назначение метки ARCH_BOOT для UEFI-загрузчикаFAT32: 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 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: 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,iocharset=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 -aAXv –exclude={«/dev/»,«/proc/*»,«/sys/*»,«/tmp/*»,«/run/*»,«/mnt/*»,«/media/*»,«/lost+found»,«/home/*/.cache/*»,«~/new_iso/*»} / ~/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 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: 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: #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 обнови 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=eth0 enp*

[Network] Address=192.168.1.150/24 Gateway=192.168.1.1

3.5 Включение сервисов (Nginx порт 7000 + SSH): 3.5.1 Правим порт Nginx: nano /etc/nginx/nginx.conf !!!!! Прописать тут что меняем (меняем listen 80 на listen 7000). 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_202605, COM-порт и отключение прерываний Hyper-V: sudo nano ~/new_iso/boot/loader/entries/arch.conf

Содержимое: ini title Arch Linux Custom Headless linux /vmlinuz-linux initrd /initramfs-linux.img options archisolabel=ARCH_202605 console=ttyS0,115200n8 earlyprintk=ttyS0,115200 rw 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 archisolabel=ARCH_202605 console=ttyS0,115200n8
earlyprintk=ttyS0,115200 root=LABEL=ARCH_ROOT rw
hv_utils.disable_gpadl_match=1
```

—— 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 5A Генерация UEFI-образа и сборка ISO через xorriso Финальный аккорд. Создаем efi.img прямо внутри структуры, чтобы xorriso собрал правильный образ. 4A.1 Создание внутреннего efi.img внутри структуры sudo dd if=/dev/zero of=~/new_iso/boot/efi.img bs=1M count=64 status=none sudo mkfs.vfat -F 16 -n «ARCH_EFI» ~/new_iso/boot/efi.img > /dev/null

mkdir -p /tmp/efi_mnt sudo mount -o loop ~/new_iso/boot/efi.img /tmp/efi_mnt 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/ sudo umount /tmp/efi_mnt

5A.2 Прямой запуск xorriso сборки гибридного ISO для Варианта А: Запускаем сборку прямо из директории ~/new_iso/. Результат положим рядом с папками в корень домашней директории. sudo xorriso -as mkisofs \

1. iso-level 3 \
2. full-iso9660-filenames \
3. volid «ARCH_202605» \
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/isohdpfx.bin \
8. eltorito-alt-boot \
9. e boot/efi.img \
10. no-emul-boot -isohybrid-gpt-basdat \
11. output ~/ARCH_202605.iso \

~/new_iso/

5А.3 Безопасное размонтирование диска донора *(Только теперь, когда ISO-образ успешно создан, освобождаем накопитель:)* Размонтируем разделы диска sdb, чтобы пересобрать их в структуру ISO. sudo umount ~/new_iso/boot sudo umount ~/new_iso/

Переходите сразу к Этапу 6.

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

Этап 5Б. Подготовка структуры и сборка ISO через xorriso

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

5Б.1 Создание чистой директории со структурой диска и копирование данных в директорию сборки. mkdir -p ~/iso_source # Синхронизируем содержимое sdb в папку для xorriso sudo rsync -aAXv ~/new_iso/ ~/iso_source/

5Б.2 Размонтируем sdb (он больше не нужен, данные у нас в iso_source) sudo umount ~/new_iso/boot sudo umount ~/new_iso

5Б.3 Создаем финальный загрузочный ISO-образ с глобальной меткой ARCH_202605. 5Б.3.1 Скрипт автоматизации сборки ISO через xorriso 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.3.1.1 Создай файл скрипта: nano ~/build_iso.sh

5.2.1.2 Вставь в него следующий код: #!/bin/bash

Настройка путей SOURCE_DIR=«$HOME/iso_source» OUTPUT_ISO=«$HOME/ARCH_202605.iso» VOLUME_ID=«ARCH_202605»

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

Монтируем efi.img и копируем туда файлы загрузчика systemd-boot mkdir -p /tmp/efi_mnt sudo mount -o loop «$SOURCE_DIR/boot/efi.img» /tmp/efi_mnt sudo mkdir -p /tmp/efi_mnt/EFI/BOOT sudo cp «$SOURCE_DIR/boot/EFI/BOOT/BOOTX64.EFI» /tmp/efi_mnt/EFI/BOOT/ sudo mkdir -p /tmp/efi_mnt/loader/entries sudo cp «$SOURCE_DIR/boot/loader/loader.conf» /tmp/efi_mnt/loader/ sudo cp «$SOURCE_DIR/boot/loader/entries/arch.conf» /tmp/efi_mnt/loader/entries/ sudo umount /tmp/efi_mnt rm -rf /tmp/efi_mnt

3. 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 «$VOLUME_ID» \
4. eltorito-boot boot/syslinux/isolinux.bin \
5. eltorito-catalog boot/syslinux/boot.cat \
6. no-emul-boot -boot-load-size 4 -boot-info-table \
7. isohybrid-mbr /usr/lib/syslinux/bios/isohdpfx.bin \
8. eltorito-alt-boot \
9. e boot/efi.img \
10. no-emul-boot -isohybrid-gpt-basdat \
11. output «$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.4 Сделай скрипт исполняемым и запусти его: 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-custom-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-custom-202605.iso. Нажми Применить (Apply) и запусти ВМ tom_2.