 Шаг 1. смотрим диски

```
lsblk sdb
```

 Шаг 2. вставить скрипт

builder.sh

```
cat << 'MAIN_EOF' > builder.sh
#!/bin/bash
set -e

# ============================================================================
# 0. ГЛОБАЛЬНАЯ ТЕХНИЧЕСКАЯ КОНФИГУРАЦИЯ СЕРВЕРА И ССЫЛОК
# ============================================================================
TARGET_FLASH="/dev/sdb"          # Физический диск флешки, размечаемый в Ubuntu
SERVER_NAME="arch-server"        # Имя хоста (hostname) будущего ПК
ADMIN_NAME="eva"                 # Логин администратора с доступом к sudo
ADMIN_PASS="65421"               # Пароль администратора
ROOT_PASS="rootpassword"         # Пароль суперпользователя (root)
OFFLINE_MODE="YES"               # Измените на "NO", если на целевом ПК есть интернет

# Базовый образ Arch Linux
URL_BOOTSTRAP="https://mirror.yandex.ru/archlinux/iso/latest/archlinux-bootstrap-x86_64.tar.zst"

# Официальные репозитории и зеркала (используются для скачивания и прописываются на ПК)
URL_YANDEX="https://mirror.yandex.ru/archlinux/\$repo/os/\$arch"
URL_PKGS="https://archlinux.pkgs.org/archlinux/\$repo/os/\$arch"
URL_RACKSPACE="http://mirror.rackspace.com/archlinux/\$repo/os/\$arch"
URL_PKGBUILD="https://geo.mirror.pkgbuild.com/\$repo/os/\$arch"

# ============================================================================
# ШАГ 1: Верификация безопасности накопителей
# ============================================================================
echo "=== ШАГ 1: Верификация безопасности накопителей ==="
UBUNTU_ROOT_DISK=$(lsblk -no PKNAME $(findmnt -nvo SOURCE /) 2>/dev/null || true)
```

```
if [ -z "$UBUNTU_ROOT_DISK" ]; then
    UBUNTU_ROOT_DISK=$(lsblk -dno NAME,MOUNTPOINTS | grep -E '/$' | awk '{print $1}')
fi

if [ "/dev/$UBUNTU_ROOT_DISK" == "$TARGET_FLASH" ]; then
    echo "КРИТИЧЕСКАЯ ОШИБКА: Запрещено размечать диск запущенной Ubuntu ($TARGET_FLASH)!"
    exit 1
fi

echo "Целевой диск определен как безопасный: $TARGET_FLASH"
echo "ВНИМАНИЕ! Диск $TARGET_FLASH будет тотально очищен через 3 секунд..."
sleep 3

# ============================================================================
# ШАГ 2: Принудительное уничтожение метаданных и разметки
# ============================================================================
echo "=== ШАГ 2: Принудительное уничтожение метаданных и разметки ==="
# Принудительно отрываем все виртуальные ФС, которые могли остаться в памяти хоста
sudo umount -l /mnt/arch/dev/pts 2>/dev/null || true
sudo umount -l /mnt/arch/dev 2>/dev/null || true
sudo umount -l /mnt/arch/proc 2>/dev/null || true
sudo umount -l /mnt/arch/sys 2>/dev/null || true
sudo umount -l /mnt/arch/boot 2>/dev/null || true
sudo umount -l /mnt/arch 2>/dev/null || true
sudo umount -l ${TARGET_FLASH}* 2>/dev/null || true

# Стираем сигнатуры файловых систем
sudo wipefs -a --force "$TARGET_FLASH"

sudo dd if=/dev/zero of="$TARGET_FLASH" bs=1M count=20 status=none
sudo partprobe "$TARGET_FLASH" || true

# ============================================================================
# ШАГ 3: Создание новой таблицы разделов  GPT
# ============================================================================
echo "=== ШАГ 3: Создание новой таблицы разделов  GPT ==="
printf "g\nn\n1\n\n+512M\nt\n1\nn\n2\n\n\nw\n" | sudo fdisk --wipe always --wipe-partitions always "$TARGET_FLASH"
sudo partprobe "$TARGET_FLASH" || true

```

```
68. if [[ "$TARGET_FLASH" == *"nvme"* ]]; then
69.     PART1="${TARGET_FLASH}p1"
70.     PART2="${TARGET_FLASH}p2"
71. else
72.     PART1="${TARGET_FLASH}1"
73.     PART2="${TARGET_FLASH}2"
74. fi
75.
76. echo "Ожидание инициализации разделов ядром..."
77. sudo partprobe /dev/sdb || true
78. sudo udevadm settle || true
79. sleep 2
80.
81. # ============================================================================
    ============
82. # ШАГ 4: Безусловное форматирование файловых систем
83. # ============================================================================
    ============
84. echo "=== ШАГ 4: Безусловное форматирование файловых систем ==="
85. sudo mkfs.vfat -F 32 "$PART1"
86. sudo mkfs.ext4 -F "$PART2"
87.
88. # ============================================================================
    ============
89. # ШАГ 5: Развертывание структуры точек монтирования
90. # ============================================================================
    ============
91. echo "=== ШАГ 5: Развертывание структуры точек монтирования ==="
92. sudo mkdir -p /mnt/arch
93. sudo mount "$PART2" /mnt/arch
94. sudo mkdir -p /mnt/arch/boot
95. sudo mount "$PART1" /mnt/arch/boot
96.
97. # ============================================================================
    ============
98. # ШАГ 6: Скачивание официального образа Arch Linux
99. # ============================================================================
    ============
100. echo "=== ШАГ 6: Скачивание официального образа Arch Linux ==="
101. sudo apt update && sudo apt install -y zstd wget
102. if [ ! -f archlinux-bootstrap-x86_64.tar.zst ]; then
103.     wget --timeout=15 --tries=3 "$URL_BOOTSTRAP"
104. fi
105. sudo tar -I zstd -xf archlinux-bootstrap-x86_64.tar.zst --strip-
     components=1 -C /mnt/arch
```

```
106.
107. #
     ================================================================
     ============
108. # ШАГ 7: Статическая генерация таблицы  fstab  флешки
109. #
     ================================================================
     ============
110. echo "=== ШАГ 7: Статическая генерация таблицы  fstab  флешки ==="
111. EFI_UUID=$(sudo blkid -s UUID -o value "$PART1")
112. ROOT_UUID=$(sudo blkid -s UUID -o value "$PART2")
113. sudo mkdir -p /mnt/arch/etc
114. printf "UUID=%s /boot vfat defaults 0 2\nUUID=%s / ext4 defaults 0
     1\n" "$EFI_UUID" "$ROOT_UUID" | sudo tee /mnt/arch/etc/fstab
115.
116. #
     ================================================================
     ============
117. # ШАГ 8: Внедрение скрипта АВТОУСТАНОВКИ на флешку (ОФФЛАЙН МЕТОД)
118. #
     ================================================================
     ============
119. echo "=== ШАГ 8: Внедрение скрипта АВТОУСТАНОВКИ на флешку ==="
120. sudo mkdir -p /mnt/arch/root
121.
122. sudo cat << EOF | sudo tee /mnt/arch/root/forest-install.sh >
     /dev/null
123. #!/bin/bash
124. set -e
125. echo "=== КОМБАЙН ЗАПУЩЁН ==="
126.
127. MY_DISK=\$(lsblk -no PKNAME \$(findmnt -nvo SOURCE /) 2>/dev/null
     || true)
128. if [ -z "\$MY_DISK" ]; then
129.     MY_DISK=\$(lsblk -no PKNAME /bootmnt 2>/dev/null || echo
     "sdb")
130. fi
131.
132. TARGET_DISK=\$(lsblk -dno NAME,TYPE | grep -v "loop" | grep -v
     "rom" | grep -v "\$MY_DISK" | head -n1 | awk '{print \$1}')
133. TARGET="/dev/\$TARGET_DISK"
134. if [ -z "\$TARGET_DISK" ]; then
135.     echo "КРИТИЧЕСКАЯ ОШИБКА: Жёсткий диск ПК не найден!"
136.     exit 1
137. fi
138.
139. echo "Целевой жёсткий диск ПК определён: \$TARGET. Уничтожаем старые
     данные..."
140. umount -f \${TARGET}* 2>/dev/null || true
141. wipefs -a --force "\$TARGET"
142. dd if=/dev/zero of="\$TARGET" bs=1M count=20 status=none
```

```
143. partprobe "\$TARGET" || true
144.
145. echo "Разметка жёсткого диска ПК..."
146. printf "g\nn\n1\n\n+512M\nt\n1\nn\n2\n\n\nw\n" | fdisk --wipe
     always --wipe-partitions always "\$TARGET"
147. if [[ "\$TARGET" == *"nvme"* ]]; then
148.     TPART1="\${TARGET}p1"
149.     TPART2="\${TARGET}p2"
150. else
151.     TPART1="\${TARGET}1"
152.     TPART2="\${TARGET}2"
153. fi
154.
155. echo "Форматирование жёсткого диска ПК..."
156. mkfs.vfat -F 32 "\$TPART1"
157. mkfs.ext4 -F "\$TPART2"
158.
159. echo "Монтирование дисков ПК под установку..."
160. mkdir -p /mnt/target && mount "\$TPART2" /mnt/target
161. mkdir -p /mnt/target/boot && mount "\$TPART1" /mnt/target/boot
162.
163. #
     ================================================================
     ============
164. # ШАГ 8.1 ОФЛАЙН УСТАНОВКА С ОБНОВЛЕНИЕМ
165. #
     ================================================================
     ============
166. #
     ================================================================
     ============
167. # НАСТОЯЩИЙ АВТОНОМНЫЙ ОФЛАЙН-МЕТОД 20.05.26 14:15
168. #
     ================================================================
     ============
169. echo "Создаем необходимые папки под локальный кэш на жестком диске ПК..."
170. mkdir -p /mnt/target/var/cache/pacman/pkg
171. mkdir -p /mnt/target/var/lib/pacman/sync
172. mkdir -p /mnt/target/etc/pacman.d
173.
174. echo "Копируем автономный кэш пакетов и базы прямо с флешки на диск ПК..."
175. if [ -d "/var/cache/pacman/pkg" ]; then
176.     cp -R /var/cache/pacman/pkg/.
     /mnt/target/var/cache/pacman/pkg/
177. fi
178. if [ -d "/var/lib/pacman/sync" ]; then
179.     cp -R /var/lib/pacman/sync/. /mnt/target/var/lib/pacman/sync/
180. fi
181.
182.
183. cp /etc/pacman.conf /mnt/target/etc/pacman.conf
```

```
touch /mnt/target/etc/pacman.d/mirrorlist

echo "Развертывание системы из локальных файлов без интернета..."
# Подставляем имена файлов через find прямо на ПК, защищая строку от Ubuntu
pacman -U --noconfirm --root /mnt/target --dbpath /mnt/target/var/lib/pacman \$(find /mnt/target/var/cache/pacman/pkg/ -name "*.pkg.tar.zst")

#
# КОНЕЦ НАСТОЯЩИЙ АВТОНОМНЫЙ ОФЛАЙН-МЕТОД 20.05.26 14:15
#

echo "Генерация таблицы fstab целевого ПК..."
printf "UUID=\$(blkid -s UUID -o value \$TPART2) / ext4 defaults 0 1\nUUID=\$(blkid -s UUID -o value \$TPART1) /boot vfat defaults 0 2\n" > /mnt/target/etc/fstab

echo "Вход в chroot окружение ПК..."
mount --types proc /proc /mnt/target/proc
mount --rbind /sys /mnt/target/sys
mount --make-rslave /mnt/target/sys
mount --rbind /dev /mnt/target/dev
mount --make-rslave /mnt/target/dev
mount --rbind /dev/pts /mnt/target/dev/pts



arch-chroot /mnt/target /bin/bash << 'CHROOT_INNER_EOF'
set -e
# Отключите автоматическую синхронизацию
# Автоматическая настройка времени без пользовательского ввода
# Вариант 1: Использовать UTC (рекомендуется)
#echo "UTC" > /etc/timezone
#ln -sf /usr/share/zoneinfo/UTC /etc/localtime
#timedatectl set-local-rtc 0

# Вариант 2: Использовать локальное время с материнки напрямую через конфиги
hwclock --systohc --localtime
mkdir -p /etc/systemd
mkdir -p /run/systemd/system
mkdir -p /run/systemd/catalog/

touch /etc/machine-id

echo -e "[Time]\nLocalRTC=yes" > /etc/adjtime
echo "Конфигурация пользователей..."
```

```
useradd -m -G wheel -s /bin/bash "${ADMIN_NAME}"

# ================================================================================
# ЖЕСТКОЕ ПОДАВЛЕНИЕ СИНЕГО ЭКРАНА НА ЦЕЛЕВОМ ПК
# ================================================================================
echo "en_US.UTF-8 UTF-8" > /etc/locale.gen
locale-gen
echo "LANG=en_US.UTF-8" > /etc/locale.conf
echo "KEYMAP=us" > /etc/vconsole.conf
echo "${SERVER_NAME}" > /etc/hostname
systemd-machine-id-setup
ln -sf /dev/null /etc/systemd/system/systemd-firstboot.service

# ================================================================================

echo "Конфигурация пользователей..."
echo "${ADMIN_NAME}:${ADMIN_PASS}" | chpasswd
echo "root:${ROOT_PASS}" | chpasswd
echo "%wheel ALL=(ALL:ALL) ALL" > /etc/sudoers.d/wheel

touch /etc/locale.conf
touch /etc/vconsole.conf
touch /etc/machine-id

echo "Активация служб..."
sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin no/' /etc/ssh/sshd_config
systemctl enable sshd NetworkManager

echo "Установка загрузчика GRUB..."
grub-install --target=x86_64-efi --efi-directory=/boot --bootloader-id=BOOT --removable --recheck
grub-mkconfig -o /boot/grub/grub.cfg
CHROOT_INNER_EOF

printf "## Russia\nServer = %s\nServer = %s\n## International\nServer = %s\nServer = %s\n" \
"${URL_YANDEX}" "${URL_PKGS}" "${URL_RACKSPACE}" "${URL_PKGBUILD}" > /mnt/target/etc/pacman.d/mirrorlist

# Размонтируем за собой виртуальные ФС на флешке
umount -l /mnt/target/dev/pts
umount -l /mnt/target/dev
umount -l /mnt/target/sys
umount -l /mnt/target/proc
```

```
umount -R /mnt/target

umount -R /mnt/target
echo "Система успешно развёрнута автономно! Перезагрузка через 3 секунды..."
sleep 3
reboot
EOF

# Настройка прав доступа к файлу автоустановщика
sudo chmod 755 /mnt/arch/root
sudo chmod +x /mnt/arch/root/forest-install.sh
sudo chmod 700 /mnt/arch/root

# ============================================================================
# ШАГ 9: Создание и регистрация фоновой службы systemd
# ============================================================================
echo "=== ШАГ 9: Создание и регистрация фоновой службы systemd ==="
sudo cat << 'SERVICE_EOF' | sudo tee /mnt/arch/etc/systemd/system/forest-autoinstall.service > /dev/null
[Unit]
Description=Автоустановщик Arch Linux без монитора
After=multi-user.target
[Service]
Type=idle
ExecStart=/root/forest-install.sh
StandardOutput=tty
StandardError=tty
[Install]
WantedBy=multi-user.target
SERVICE_EOF

sudo mkdir -p /mnt/arch/etc/systemd/system/multi-user.target.wants
sudo ln -sf /etc/systemd/system/forest-autoinstall.service /mnt/arch/etc/systemd/system/multi-user.target.wants/forest-autoinstall.service

# ============================================================================
# ШАГ 10: Формирование чистых репозиториев для флешки
# ============================================================================
echo "=== ШАГ 10: Формирование чистых репозиториев для флешки ==="
sudo mkdir -p /mnt/arch/etc/pacman.d

```

```
306. # Пишем зеркала через стандартный echo, полностью избегая капризных
     Here-Doc с пайпами
307. echo "Server = ${URL_YANDEX}" | sudo tee
     /mnt/arch/etc/pacman.d/mirrorlist > /dev/null
308. echo "Server = ${URL_PKGS}" | sudo tee -a
     /mnt/arch/etc/pacman.d/mirrorlist > /dev/null
309.
310. #
     ============================================================================
311. # ШАГ 11: Подготовка локального репозитория на флешке (Выполняет Ubuntu)
312. #
     ============================================================================
313. echo "=== ШАГ 11: Подготовка локального репозитория силами Ubuntu ==="
314. sudo mkdir -p /mnt/arch/var/cache/pacman/pkg
315. sudo mkdir -p /mnt/arch/var/lib/pacman
316.
317. # Генерируем правильный pacman.conf для флешки (пока со стандартными
     зеркалами)
318. cat << 'EOF' | sudo tee /mnt/arch/etc/pacman.conf > /dev/null
319. [options]
320. Architecture = auto
321. SigLevel = Optional TrustAll
322. LocalFileSigLevel = Optional
323.
324. [core]
325. Include = /etc/pacman.d/mirrorlist
326.
327. [extra]
328. Include = /etc/pacman.d/mirrorlist
329. EOF
330.
331. # Скачиваем базы данных и ВСЕ пакеты, зайдя в chroot флешки (тут есть
     библиотеки и интернет)
332. sudo /mnt/arch/usr/bin/arch-chroot /mnt/arch /bin/bash <<
     'CHROOT_EOF'
333.
334. set -e
335.
336. echo "Инициализация ключей внутри окружения флешки..."
337. pacman-key --init
338. pacman-key --populate archlinux
339.
340. echo "Скачиваем базы данных и ВСЕ пакеты в автономный кэш флешки..."
341. pacman -Syw --noconfirm \
342.     iana-etc filesystem linux-api-headers tzdata licenses \
343. glibc libgcc libstdc++ libasan libatomic libgfortran libgomp \
344. liblsan libobjc libquadmath libtsan libubsan gcc-libs \
345. ncurses readline bash acl attr gmp zlib sqlite  \
346. util-linux-libs e2fsprogs keyutils gdbm brotli xz \
```

```
lz4 zstd openssl libsasl libldap libevent libverto lmdb \
krb5 libcap-ng audit libxcrypt libtirpc libnsl pambase \
libgpg-error libgcrypt systemd-libs pam libcap coreutils \
bzip2 libseccomp file findutils mpfr gawk pcre2 grep \
procps-ng sed tar libtasn1 libffi libp11-kit p11-kit \
ca-certificates-utils ca-certificates-mozilla ca-certificates \
libunistring libidn2 libnghttp2 libnghttp3 nettle leancrypto \
gnutls libngtcp2 libpsl libssh2 curl json-c gnulib-l10n icu \
libxml2 gettext hwdata kmod pciutils psmisc shadow util-linux \
gzip licenses libksba libusb libassuan libsysprof-capture \
glib2 tpm2-tss libsecret pinentry npth gnupg gpgme libarchive \
pacman-mirrorlist device-mapper popt cryptsetup expat \
dbus dbus-broker dbus-broker-units dbus-units kbd libelf \
systemd jansson binutils libmakepkg-dropins pacman \
archlinux-keyring systemd-sysvcompat iputils libmnl \
libnfnetlink libnetfilter_conntrack libnftnl libnl libpcap \
nftables iptables libbpf iproute2 base mkinitcpio-busybox \
diffutils mkinitcpio \
linux linux-firmware-whence linux-firmware-amdgpu \
linux-firmware-atheros linux-firmware-broadcom \
linux-firmware-cirrus linux-firmware-intel \
linux-firmware-mediatek linux-firmware-nvidia \
linux-firmware-other linux-firmware-radeon \
linux-firmware-realtek linux-firmware libedit \
openssh libmm-glib libndp gpm pcre slang libnewt nspr nss \
libnm libdaemon libsodium libpgm zeromq libteam \
mobile-broadband-provider-info duktape polkit \
pcsclite wpa_supplicant networkmanager sudo \
nano dosfstools mtools mailcap nginx libmd libbsd \
talloc tevent tdb ldb avahi libcups liburing libwbclient \
mpdecimal python cifs-utils smbclient samba grub efivar \
efibootmgr libzip argon2 oniguruma php php-fpm



CHROOT_EOF

# Переводим pacman.conf флешки на полностью локальный оффлайн-режим file:///
cat << 'EOF' | sudo tee /mnt/arch/etc/pacman.conf > /dev/null
[options]
Architecture = auto
SigLevel = Optional TrustAll
LocalFileSigLevel = Optional

[core]
Server = file:///var/cache/pacman/pkg

[extra]
```

```
Server = file:///var/cache/pacman/pkg
EOF


# ================================================================================
# ШАГ 12: Локальная автономная установка на флешку (БЕЗ ИНТЕРНЕТА)
# ================================================================================
echo "=== ШАГ 12: Накатывание базовой системы на флешку из локального кэша ==="

# Генерируем базовые конфиги флешки до chroot, чтобы заблокировать синий экран
echo "en_US.UTF-8 UTF-8" | sudo tee /mnt/arch/etc/locale.gen > /dev/null
echo "LANG=en_US.UTF-8" | sudo tee /mnt/arch/etc/locale.conf > /dev/null
echo "KEYMAP=us" | sudo tee /mnt/arch/etc/vconsole.conf > /dev/null
echo "arch-flash" | sudo tee /mnt/arch/etc/hostname > /dev/null

# Принудительная инициализация идентификаторов системы
sudo systemd-machine-id-setup --root=/mnt/arch
sudo touch /mnt/arch/etc/machine-id

# ЖЕСТКОЕ ОФФЛАЙН ПОДАВЛЕНИЕ FIRSTBOOT (Записываем строго внутрь флешки с sudo)
sudo mkdir -p /mnt/arch/etc/systemd/system
sudo ln -sf /dev/null /mnt/arch/etc/systemd/system/systemd-firstboot.service
sudo touch /mnt/arch/var/lib/systemd/clock

# РЕШЕНИЕ ПРОБЛЕМЫ 3/13 И 12/13: Создаем каталоги и монтируем виртуальные ФС ядра хоста
sudo mkdir -p /mnt/arch/usr/lib/systemd/catalog

echo "Развертываем систему на флешке из автономного кэша..."
# Используем встроенный chroot флешки, передавая команды через bash -c, чтобы не ломать /dev/stdin
sudo /mnt/arch/usr/bin/arch-chroot /mnt/arch /bin/bash -c "
set -e
# Ставим напрямую по маске файлы пакетов
pacman -U --noconfirm /var/cache/pacman/pkg/*.pkg.tar.zst

# Настраиваем GRUB на флешке как съемный диск
grub-install --target=x86_64-efi --efi-directory=/boot --bootloader-id=Arch_Flash --removable --recheck
grub-mkconfig -o /boot/grub/grub.cfg
```

```
"
#
================================================================================
# ШАГ 13: финальная проверка
#
================================================================================
echo "=== ШАГ 13: финальная проверка ==="

sudo umount -l /mnt/arch/boot 2>/dev/null || true
sudo umount -l /mnt/arch 2>/dev/null || true


MAIN_EOF
```

 Шаг 4. Очистите его от символов Windows

```
sed -i 's/\r$//' builder.sh
```

 Шаг 5. Разрешите запуск

```
chmod +x builder.sh && ./builder.sh
```

 Шаг 4. запустить скрипт

```
./builder.sh
```

From:
https://wwoss.ru/ - **worldwide open-source software**

Permanent link:
https://wwoss.ru/doku.php?id=tmp

Last update: **2026/05/22 12:15**