<link rel=«preconnect» href=«*wwoss.ru:443*»> *<link rel=«dns-prefetch» href=*«wwoss.ru:443»> <!– Start Open Web Analytics Tracker –> <script type=«text/javascript»> //<![CDATA[var owa_baseUrl = 'https://wwoss.ru:443/analytics/'; var owa_cmds = owa_cmds || []; owa_cmds.push(['setSiteId', 'OWA-53ed90016869004b']); owa_cmds.push(['trackPageView']); owa_cmds.push(['trackClicks']);

(function() {

```
  var _owa = document.createElement('script'); _owa.async = true;
  owa_baseUrl = ('https:' == document.location.protocol ?
window.owa_baseSecUrl || owa_baseUrl.replace(/http:/, 'https:') :
owa_baseUrl );
  _owa.src = owa_baseUrl + 'public/base/dist/owa.tracker.js';
  var _owa_p = document.head || document.getElementsByTagName('head')[0] ||
document.documentElement;
  _owa_p.appendChild(_owa);
```

}()); //]]> </script> <!– End Open Web Analytics Code –> □ 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)
```

```
19. URL_YANDEX="https://mirror.yandex.ru/archlinux/$repo/os/$arch"
20. URL_PKGS="https://archlinux.pkgs.org/archlinux/$repo/os/$arch"
21. URL_RACKSPACE="http://mirror.rackspace.com/archlinux/$repo/os/$arc
    h"
22. URL_PKGBUILD="https://geo.mirror.pkgbuild.com/$repo/os/$arch"
23.
24. #
    ================================================================
    ============
25. # ШАГ 1: Верификация безопасности накопителей
26. #
    ================================================================
    ============
27. echo "=== ШАГ 1: Верификация безопасности накопителей ==="
28. UBUNTU_ROOT_DISK=$(lsblk -no PKNAME $(findmnt -nvo SOURCE /)
    2>/dev/null || true)
29. if [ -z "$UBUNTU_ROOT_DISK" ]; then
30.     UBUNTU_ROOT_DISK=$(lsblk -dno NAME,MOUNTPOINTS | grep -E '/$'
    | awk '{print $1}')
31. fi
32.
33. if [ "/dev/$UBUNTU_ROOT_DISK" == "$TARGET_FLASH" ]; then
34.     echo "КРИТИЧЕСКАЯ ОШИБКА: Запрещено размечать диск запущенной
    Ubuntu ($TARGET_FLASH)!"
35.     exit 1
36. fi
37.
38. echo "Целевой диск определен как безопасный: $TARGET_FLASH"
39. echo "ВНИМАНИЕ! Диск $TARGET_FLASH будет тотально очищен через 5 секунд..."
40. sleep 5
41.
42. #
    ================================================================
    ============
43. # ШАГ 2: Принудительное уничтожение метаданных и разметки
44. #
    ================================================================
    ============
45. echo "=== ШАГ 2: Принудительное уничтожение метаданных и разметки ==="
46. sudo umount -f ${TARGET_FLASH}* 2>/dev/null || true
47. sudo wipefs -a --force "$TARGET_FLASH"
48. sudo dd if=/dev/zero of="$TARGET_FLASH" bs=1M count=20 status=none
49. sudo partprobe "$TARGET_FLASH" || true
50.
51. #
    ================================================================
    ============
52. # ШАГ 3: Создание новой таблицы разделов  GPT
53. #
    ================================================================
    ============
```

```
54. echo "=== ШАГ 3: Создание новой таблицы разделов GPT ==="
55. printf "gnnn1nn+512Mntn1nnn2nnnwn" | sudo fdisk --wipe always --
    wipe-partitions always "$TARGET_FLASH"
56. sudo partprobe "$TARGET_FLASH" || true
57.
58. if [[ "$TARGET_FLASH" == *"nvme"* ]]; then
59.     PART1="${TARGET_FLASH}p1"
60.     PART2="${TARGET_FLASH}p2"
61. else
62.     PART1="${TARGET_FLASH}1"
63.     PART2="${TARGET_FLASH}2"
64. fi
65.
66. echo "Ожидание инициализации разделов ядром..."
67. sudo partprobe /dev/sdb || true
68. sudo udevadm settle || true
69. sleep 2
70.
71. #
    ============================================================================
    ============
72. # ШАГ 4: Безусловное форматирование файловых систем
73. #
    ============================================================================
    ============
74. echo "=== ШАГ 4: Безусловное форматирование файловых систем ==="
75. sudo mkfs.vfat -F 32 "$PART1"
76. sudo mkfs.ext4 -F "$PART2"
77.
78. #
    ============================================================================
    ============
79. # ШАГ 5: Развертывание структуры точек монтирования
80. #
    ============================================================================
    ============
81. echo "=== ШАГ 5: Развертывание структуры точек монтирования ==="
82. sudo mkdir -p /mnt/arch
83. sudo mount "$PART2" /mnt/arch
84. sudo mkdir -p /mnt/arch/boot
85. sudo mount "$PART1" /mnt/arch/boot
86.
87. #
    ============================================================================
    ============
88. # ШАГ 6: Скачивание официального образа Arch Linux
89. #
    ============================================================================
    ============
90. echo "=== ШАГ 6: Скачивание официального образа Arch Linux ==="
91. sudo apt update && sudo apt install -y zstd wget
```

```
92. if [ ! -f archlinux-bootstrap-x86_64.tar.zst ]; then
93.     wget --timeout=15 --tries=3 "$URL_BOOTSTRAP"
94. fi
95. sudo tar -I zstd -xf archlinux-bootstrap-x86_64.tar.zst --strip-
    components=1 -C /mnt/arch
96.
97. #
    ============================================================================
    ============
98. # ШАГ 7: Статическая генерация таблицы fstab флешки
99. #
    ============================================================================
    ============
100. echo "=== ШАГ 7: Статическая генерация таблицы fstab флешки ==="
101. EFI_UUID=$(sudo blkid -s UUID -o value "$PART1")
102. ROOT_UUID=$(sudo blkid -s UUID -o value "$PART2")
103. sudo mkdir -p /mnt/arch/etc
104. printf "UUID=%s /boot vfat defaults 0 2nUUID=%s / ext4 defaults 0
     1n" "$EFI_UUID" "$ROOT_UUID" | sudo tee /mnt/arch/etc/fstab
105.
106. #
     ============================================================================
     ============
107. # ШАГ 8: Внедрение скрипта АВТОУСТАНОВКИ на флешку (ОФФЛАЙН МЕТОД)
108. #
     ============================================================================
     ============
109. echo "=== ШАГ 8: Внедрение скрипта АВТОУСТАНОВКИ на флешку ==="
110. sudo mkdir -p /mnt/arch/root
111.
112. sudo cat << EOF | sudo tee /mnt/arch/root/forest-install.sh >
     /dev/null
113. #!/bin/bash
114. set -e
115. echo "=== КОМБАЙН ЗАПУЩЁН ==="
116.
117. MY_DISK=$(lsblk -no PKNAME $(findmnt -nvo SOURCE /) 2>/dev/null ||
     true)
118. if [ -z "$MY_DISK" ]; then
119.     MY_DISK=$(lsblk -no PKNAME /bootmnt 2>/dev/null || echo "sdb")
120. fi
121.
122. TARGET_DISK=$(lsblk -dno NAME,TYPE | grep -v "loop" | grep -v
     "rom" | grep -v "$MY_DISK" | head -n1 | awk '{print $1}')
123. TARGET="/dev/$TARGET_DISK"
124. if [ -z "$TARGET_DISK" ]; then
125.     echo "КРИТИЧЕСКАЯ ОШИБКА: Жёсткий диск ПК не найден!"
126.     exit 1
127. fi
128.
129. echo "Целевой жёсткий диск ПК определён: $TARGET. Уничтожаем старые
```

```
    данные..."
umount -f ${TARGET}* 2>/dev/null || true
wipefs -a --force "$TARGET"
dd if=/dev/zero of="$TARGET" bs=1M count=20 status=none
partprobe "$TARGET" || true

echo "Разметка жёсткого диска ПК..."
printf "gnnn1nn+512Mntn1nnn2nnnwn" | fdisk --wipe always --wipe-
    partitions always "$TARGET"
if [[ "$TARGET" == *"nvme"* ]]; then
    TPART1="${TARGET}p1"
    TPART2="${TARGET}p2"
else
    TPART1="${TARGET}1"
    TPART2="${TARGET}2"
fi

echo "Форматирование жёсткого диска ПК..."
mkfs.vfat -F 32 "$TPART1"
mkfs.ext4 -F "$TPART2"

echo "Монтирование дисков ПК под установку..."
mkdir -p /mnt/target && mount "$TPART2" /mnt/target
mkdir -p /mnt/target/boot && mount "$TPART1" /mnt/target/boot

#
    ============================================================================
    ============
# ШАГ 8.1 ОФЛАЙН УСТАНОВКА С ОБНОВЛЕНИЕМ
#
    ============================================================================
    ============
if [ "${OFFLINE_MODE}" == "YES" ]; then
    echo "=== ВЫБРАН РЕЖИМ: ТОТАЛЬНЫЙ ОФЛАЙН ==="
    mkdir -p /mnt/target/var/cache/pacman/pkg
    mkdir -p /mnt/target/var/lib/pacman

    echo "Копируем автономный кэш пакетов с флешки на диск ПК..."
    # Пишем команду так, чтобы в ней не было круглых скобок $() и  find
    cp -R /var/cache/pacman/pkg/.
    /mnt/target/var/cache/pacman/pkg/

    echo "Переводим системный  pacman  флешки в автономный режим..."
    # Подменяем системный файл зеркал флешки, указывая вместо сайтов
    локальный путь  file://
    mkdir -p /etc/pacman.d
    echo "Server = file:///var/cache/pacman/pkg" >
    /etc/pacman.d/mirrorlist

    echo "Развертывание системы из локальных файлов без интернета..."
    # Теперь  pacman -Syu  прочитает родной конфиг, увидит тамfile:// и
```

```
    молча поставит пакеты из папки
172.     pacman -Syu --noconfirm --root /mnt/target --dbpath
    /mnt/target/var/lib/pacman base linux linux-firmware openssh
    networkmanager sudo nano dosfstools mtools samba grub efibootmgr
    nginx php php-fpm
173. else
174. #
    ================================================================
    ============
175. # ШАГ 8.2 ОНЛАЙН УСТАНОВКА С ОБНОВЛЕНИЕМ
176. #
    ================================================================
    ============
177.     echo "=== ВЫБРАН РЕЖИМ: ОНЛАЙН УСТАНОВКА С ОБНОВЛЕНИЕМ ==="
178.  echo "Создаем чистые папки под кэш на жестком диске ПК..."
179.     mkdir -p /mnt/target/var/cache/pacman/pkg
180.     mkdir -p /mnt/target/var/lib/pacman
181.     mkdir -p /mnt/target/etc/pacman.d
182.
183.     echo "Записываем готовый адрес зеркала Яндекса в целевой ПК..."
184.     # Переменная URL_YANDEX уже содержит идеальную ссылку, просто
    убираем один лишний слэш перед arch
185.     echo "Server = ${URL_YANDEX/\\\$arch/\$arch}" >
    /mnt/target/etc/pacman.d/mirrorlist
186.
187.     # Записываем полученный адрес зеркала в зеркала целевого ПК
188.     echo "Server = ${URL_YANDEX}" >
    /mnt/target/etc/pacman.d/mirrorlist
189.     echo "Server = ${URL_PKGS}" >>
    /mnt/target/etc/pacman.d/mirrorlist
190.     echo "Server = ${URL_RACKSPACE}/\$repo/os/$arch" >>
    /mnt/target/etc/pacman.d/mirrorlist
191.     echo "Server = ${URL_PKGBUILD}/\$repo/os/$arch" >>
    /mnt/target/etc/pacman.d/mirrorlist
192.
193.     # Прописываем DNS Google прямо в сетевой конфиг запущенной флешки
194.     echo "nameserver 8.8.8.8" > /etc/resolv.conf
195.     echo "nameserver 1.1.1.1" >> /etc/resolv.conf
196.
197.     cat << 'PAC_EOF' > /tmp/pacman-install.conf
198.     [options]
199.     Architecture = auto
200.     SigLevel = Required DatabaseOptional
201.     LocalFileSigLevel = Optional
202.
203.     [core]
204.     Include = /mnt/target/etc/pacman.d/mirrorlist
205.
206.     [extra]
207.     Include = /mnt/target/etc/pacman.d/mirrorlist
208. PAC_EOF
```

```
209.
210.     echo "Принудительно включаем DNS-серверы на запущенной флешке..."
211.     cp /etc/resolv.conf /etc/resolv.conf.bak && echo "nameserver
     8.8.8.8" > /etc/resolv.conf
212.
213.     echo "Скачиваем свежие базы репозиториев напрямую на жесткий диск
     ПК..."
214.     # ИСПРАВЛЕНИЕ: Этот шаг скачает файлы core.db и extra.db прямо на
     HDD компьютера
215.     pacman -Sy --noconfirm --config /tmp/pacman-install.conf --
     dbpath /mnt/target/var/lib/pacman
216.
217.     echo "Скачиваем свежие пакеты из интернета в кэш жесткого диска ПК..."
218.     pacman -Syw --noconfirm --config /tmp/pacman-install.conf --
     cachedir /mnt/target/var/cache/pacman/pkg --dbpath
     /mnt/target/var/lib/pacman \
219.     base linux linux-firmware openssh networkmanager sudo nano
     dosfstools mtools samba grub efibootmgr nginx php php-fpm
220.
221.     echo "Развертывание системы из скачанных из интернета файлов..."
222.     # Копируем созданный рабочий зеркальный лист в саму систему, чтобы она
     знала зеркала после перезагрузки
223.     cp /mnt/target/etc/pacman.d/mirrorlist
     /mnt/target/etc/pacman.d/mirrorlist.bak
224.
225.     pacman -U --noconfirm --root /mnt/target --dbpath
     /mnt/target/var/lib/pacman
     /mnt/target/var/cache/pacman/pkg/*.pkg.tar.zst
226. fi
227. #
     ================================================================
     ============
228.
229. umount /mnt/target/var/cache/pacman/pkg
230.
231. echo "Генерация таблицы fstab целевого ПК..."
232. 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
233.
234. echo "Вход в chroot окружение ПК..."
235. /mnt/arch/bin/arch-chroot /mnt/target /bin/bash << 'CHROOT_EOF'
236. set -e
237. # Отключите автоматическую синхронизацию
238. # Автоматическая настройка времени без пользовательского ввода
239. # Вариант 1: Использовать UTC (рекомендуется)
240. #echo "UTC" > /etc/timezone
241. #ln -sf /usr/share/zoneinfo/UTC /etc/localtime
242. #timedatectl set-local-rtc 0
243.
244. # Вариант 2: Использовать локальное время с материнки
```

```
hwclock --systohc
timedatectl set-local-rtc 1
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
sudo ln -sf /dev/null /mnt/arch/etc/systemd/system/systemd-firstboot.service
# ============================================================================

echo "Конфигурация пользователей..."
useradd -m -G wheel -s /bin/bash "${ADMIN_NAME}"
echo "${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=Arch_PC --recheck
grub-mkconfig -o /boot/grub/grub.cfg
CHROOT_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

```

```
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

    cat << MIRROR_EOF | sudo tee /mnt/arch/etc/pacman.d/mirrorlist > /dev/null
```

```
## Russia
Server = ${URL_YANDEX}
Server = ${URL_PKGS}

## International
Server = ${URL_RACKSPACE}
Server = ${URL_PKGBUILD}
MIRROR_EOF

# ============================================================================
# ШАГ 11: Подготовка локального репозитория на флешке
# ============================================================================
echo "=== ШАГ 11: Подготовка локального репозитория ==="
sudo mkdir -p /mnt/arch/var/cache/pacman/pkg
sudo mkdir -p /mnt/arch/var/lib/pacman

sudo mkdir -p /mnt/arch/etc/pacman.d
echo "Server = ${URL_YANDEX//\\\\\\$/\$}" | sudo tee /mnt/arch/etc/pacman.d/mirrorlist > /dev/null



    cat << 'EOF' | sudo tee /mnt/arch/etc/pacman.conf > /dev/null
    [options]
    Architecture = auto
    SigLevel = Required DatabaseOptional
    LocalFileSigLevel = Optional

    [core]
    Include = /etc/pacman.d/mirrorlist

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

# ============================================================================
# ШАГ 12: Установка пакетов на флешку и кэширование пакетов для ПК
# ============================================================================
echo "=== ШАГ 12: Накатывание пакетов и загрузчика на флешку ==="

echo "nameserver 8.8.8.8" | sudo tee /mnt/arch/etc/resolv.conf

# ДОБАВЛЕНО: Генерируем базовые конфиги флешки до chroot, чтобы
```

```
    заблокировать синий экран systemd-firstboot
echo "en_US.UTF-8 UTF-8" | sudo tee /mnt/arch/etc/locale.gen
echo "LANG=en_US.UTF-8" | sudo tee /mnt/arch/etc/locale.conf
echo "KEYMAP=us" | sudo tee /mnt/arch/etc/vconsole.conf
echo "arch-flash" | sudo tee /mnt/arch/etc/hostname
sudo systemd-machine-id-setup --root=/mnt/arch

# ТОТ САМЫЙ КЛЮЧЕВОЙ КОСТЫЛЬ: Жестко блокируем запуск синего экрана на
    самой флешке
sudo ln -sf /dev/null /mnt/arch/etc/systemd/system/systemd-
    firstboot.service

# Возвращаем пути к зеркалам в стандартный вид для работы внутри самой
    флешки
sudo sed -i 's|/mnt/arch||g' /mnt/arch/etc/pacman.conf

sudo /mnt/arch/bin/arch-chroot /mnt/arch /bin/bash << 'EOF'
set -e
pacman-key --init
pacman-key --populate archlinux
# Обновляем базы пакетов
# Объединяем обновление баз и скачивание пакетов в ОДНУ команду
pacman -Syuw --noconfirm base linux linux-firmware openssh
    networkmanager sudo nano dosfstools mtools nginx samba grub
    efibootmgr
# Устанавливаем систему и загрузчик на флешку
pacman -S --noconfirm base linux linux-firmware openssh
    networkmanager sudo nano dosfstools mtools nginx grub efibootmgr
# Принудительно генерируем локали
locale-gen
# 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
EOF

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

# Размонтируем каталоги, если они еще примонтированы
sudo umount -l /mnt/arch/dev || true
sudo umount -l /mnt/arch/proc || true
sudo umount -l /mnt/arch/sys || true
sudo umount -l /mnt/arch || true

```

```
404.
405. 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_20.05.26&rev=1791893871

Last update: **2026/10/13 15:17**