Archiso (Русский)

Эта страница нуждается в сопроводителе

Статья не гарантирует актуальность информации. Помогите русскоязычному сообществу поддержкой подобных страниц. См. **Команда переводчиков ArchWiki**

Contents

- 1Что это?
- 2Установка Archiso
- 3Конфигурация создаваемой системы
  - 3.1Установка пакетов
    - 3.1.1Пользовательский локальный репозиторий
  - 3.2Makefile
  - 3.3mkinitcpio.conf
  - 3.4packages.list
  - 3.5isomounts
  - 3.6boot-files
    - 3.6.1Использование isolinux
  - 3.7overlay
    - 3.7.1Добавление fstab
    - 3.7.2Добавление пользователя
      - 3.7.2.1Вручную
      - 3.7.2.2Используя useradd
    - 3.7.3Добавление чего-либо в домашнюю директорию пользователя при загрузке
    - 3.7.4Закрытие темы оверлея
- 4Генерация образа
- 5Links

Что это?

Archiso - набор bash скриптов, предназначенных для создания полностью функциональных Live-CD/DVD и Live-USB на базе Arch Linux. Это довольно гибкий инструмент, который может быть использован как для создания дисков восстановления или установочных, так и для специализированных live-CD/DVD/USB систем. Центр Archiso - **mkarchiso**. Для получения подробного описания всех его опций достаточно вызвать его без параметров, так что здесь будет описанно только создание live диска своими руками.

Благодаря последним изменениям Archiso создаёт ISO образы, пригодные как для записи на диск, так и на флеш носитель.

Установка Archiso

Примечание: Archiso должен быть установлен для платформы x86_64. [1] В противном случае, скорее всего, возможны проблемы, решение которых будет рассмотрено дальше.

Для начала нужно установить пакет **archiso** или **archiso-git**[AUR].

Archiso поставляется с двумя профилями "profiles": *releng* и *baseline*.

- Если вы хотите создать полностью индивидуальную версию Arch Linux, предварительно установленной со всеми вашими любимыми программами и конфигурациями, используйте профиль "releng".

- Если вы просто хотите создать оптимальную для существования, без предварительно установленных пакетов и минимальной конфигурацией, используйте "baseline".

Теперь скопируйте профиль на Ваш выбор, в каталог, в котором вы можете вносить корректировки (мы будем использовать `~/archlive`). Выполните следующую команду,

заменив **`profile`** на `releng` или `baseline`.

```
# cp -r /usr/share/archiso/configs/profile/* ~/archlive
```

- Если вы используете `releng` профиль, создавая полностью индивидуальный образ, вы можете переходить к **#Настройка Live носителя**.
- Если вы используете `baseline` профиль для создания голого образа, у вас не будет времени на настройку и можно переходить на **#Создать ISO**.

Конфигурация создаваемой системы

В этом разделе подробно описывается настройка создаваемого вами образа, определение пакетов и конфигурация. Внутри каталога `~/archlive` есть ряд файлов и каталогов; мы рассмотрим лишь несколько из них, в основном:

- `packages.*` - это где вы перечисляете построчно пакеты, которые вы хотите установить, и
- `airootfs` каталог - это каталог выступает в качестве оверлея, это где вы делаете все настройки.

Как правило, любые административные задачи, которые вы обычно делаете после новой установки, могут быть выполнены в скрипте `~/archlive/airootfs/root/customize_airootfs.sh`, за исключением установки пакетов. Скрипт должен быть написан с точки зрения новой среды, поэтому `/` в скрипте означает корень ISO-образа, который создается.

Установка пакетов

Желаете создать список пакетов, которые хотите установить на вашу систему прямо с компакт-диска. Формат файл содержит полные названия пакетов, каждый пакет с новой строки. Это очень удобно для создания собственного интереса LiveCD, просто укажите нужные пакеты и создавайте образ. Файлы packages.i686 и packages.x86_64 позволяют установить программное обеспечение на только 32-бит или 64-бит систему соответственно.

Примечание: Если вы хотите использовать **оконный менеджер** в Live CD, то вы должны добавить необходимые и правильные **видео драйвера**, или WM может зависнуть при загрузке.

Пользовательский локальный репозиторий

This article or section is a candidate for merging with Pacman tips#Custom local repository.

Notes: Move the general information (e.g. repo tree) into the main article. (Discuss in **Talk:Archiso (Русский)#**)

Вы также можете **создать пользовательский локальный репозиторий** в целях подготовки пакетов из **AUR** или **ABS**. При этом пакеты для обеих архитектур, должны следовать в определенном порядке каталогов, чтобы не столкнуться с проблемами. Например:

- `~/customrepo`
  - `~/customrepo/x86_64`
    - `~/customrepo/x86_64/foo-x86_64.pkg.tar.xz`
    - `~/customrepo/x86_64/customrepo.db.tar.gz`
    - `~/customrepo/x86_64/customrepo.db` (симлинк создается `repo-add`)
  - `~/customrepo/i686`

- `~/customrepo/i686/foo-i686.pkg.tar.xz`
- `~/customrepo/i686/customrepo.db.tar.gz`
- `~/customrepo/i686/customrepo.db` (симлинк создается `repo-add`)

Затем вы можете добавить ваш репозиторий внеся изменения в файле `~/archlive/pacman.conf`, поставив его над другой записью репозитория (для более высшего приоритета):

```
# custom repository
[customrepo]
SigLevel = Optional TrustAll
Server = file:///home/user/customrepo/$arch
```

Makefile

Первым делом создаём отдельную директорию для работы и переходим в неё.

```
$ mkdir myarch && cd myarch
```

Далее создаём Makefile и прописываем (или обдуманно копипастим от сюда) туда нижеследующие инструкции.

```
$ vim Makefile
```

Ниже вы найдёте пример Makefile.

Важно: Все отступы в файле делать табуляцией. На пробелы будет ругаться.

Важно: В ранних версиях работала команда "mkinitcpio -c mkinitcpio.conf..." так или иначе (возможно, ввиду последних обновлений) теперь, если скрипт не может найти конфиг-файл, указывайте к последнему полный путь в mkinitcpio.conf, иначе вы можете получить пустой initcpio на выходе, из-за чего полученная система не будет загружаться (VFS error, kernel panic)

```
#### Редактируйте данный файл для модифицирования конечной системы.
#  Рабочая директория для построения системы.
WORKDIR=work
#  Список устанавливаемых приложений, either space separated in a string or
line separated in a file. Может включать группы.
PACKAGES="$(shell cat packages.list) syslinux"
# Имя дистрибутива. Не зависит от/не определяет целевую архитектуру.
NAME=myarch
# Версия дистрибутива.
VER=1.00
# Версия ядра.
KVER=2.6.32-ARCH
# Архитектура.
```

```
ARCH?=$(shell uname -m)
# Директория, в которой находился пользователь, запустивший скрипт
PWD:=$(shell pwd)
# Полное (финальное) имя образа.
FULLNAME="$(PWD)"/$(NAME)-$(VER)-$(ARCH)

# Умолчальная инструкция make'у, для компиляции всего(?) (оригинал:"Default make instruction to build everything.")
all: myarch

# Запуск base-fs перед сборкой финального ISO образа.
myarch: base-fs
	mkarchiso -p syslinux iso "$(WORKDIR)" "$(FULLNAME)".iso

# Основное правило для процесса создания файловой системы образа. Приложения отрабатывают слева на право.
# Тоесть, сначала root-image в конце - syslinux.
base-fs: root-image boot-files initcpio overlay iso-mounts syslinux

# root-image всегда запускается первым.
# Скачивание и установка приложений в $WORKDIR.
root-image: "$(WORKDIR)"/root-image/.arch-chroot
"$(WORKDIR)"/root-image/.arch-chroot:
root-image:
	mkarchiso -p $(PACKAGES) create "$(WORKDIR)"

# Правило для создания /boot
boot-files: root-image
	cp -r "$(WORKDIR)"/root-image/boot "$(WORKDIR)"/iso/
	cp -r boot-files/* "$(WORKDIR)"/iso/boot/

# Правило для образов initcpio
initcpio: "$(WORKDIR)"/iso/boot/myarch.img
"$(WORKDIR)"/iso/boot/myarch.img: mkinitcpio.conf "$(WORKDIR)"/root-image/.arch-chroot
	mkdir -p "$(WORKDIR)"/iso/boot
	mkinitcpio -c ./mkinitcpio.conf -b "$(WORKDIR)"/root-image -k $(KVER) -g $@
```

```
# Подробнее см.: Overlay
overlay:
	mkdir -p "$(WORKDIR)"/overlay/etc/pacman.d
	cp -r overlay "$(WORKDIR)"/
	wget -O "$(WORKDIR)"/overlay/etc/pacman.d/mirrorlist
https://www.archlinux.org/mirrorlist/all/
	sed -i "s/#Server/Server/g"
"$(WORKDIR)"/overlay/etc/pacman.d/mirrorlist

# Правило для создания isomounts.
iso-mounts: "$(WORKDIR)"/isomounts
"$(WORKDIR)"/isomounts: isomounts root-image
	sed "s|@ARCH@|$(ARCH)|g" isomounts > $@

# Исполняется перед генерацией финального образа.
syslinux: root-image
	mkdir -p $(WORKDIR)/iso/boot/isolinux
	cp $(WORKDIR)/root-image/usr/lib/syslinux/*.c32
$(WORKDIR)/iso/boot/isolinux/
	cp $(WORKDIR)/root-image/usr/lib/syslinux/isolinux.bin
$(WORKDIR)/iso/boot/isolinux/

# При вызове "make clean" отчищает систему от вчего, созданного в процессе
создания образа.
clean:
	rm -rf "$(WORKDIR)" "$(FULLNAME)".img "$(FULLNAME)".iso

.PHONY: all myarch
.PHONY: base-fs
.PHONY: root-image boot-files initcpio overlay iso-mounts
.PHONY: syslinux
.PHONY: clean
```

Тоесть, при исполнении "make myarch" из под рута,происходит следующее:

- **root-image** скачивает и устанавливает выбранные приложения в *$WORKDIR*
- **boot-files** готовит загрузочные файлы и копирует загрузочные скрипты
- **initcpio** работает с *initcpio*
- **overlay** копирует файлы, перекрывающие базовую конфигурацию в *root-image* в *$WORKDIR*
- **iso-mounts** немного уличной магии с применением sed, чтобы AUFS знала, куда ей монтироваться при загрузке
- **syslinux** копирует загрузчик
- **myarch** создаёт конечный образ, пригодный для записи на CD/DVD/флешку.

Одного Makefile будет недостаточно, так что нужно будет создать файлы, описанные ниже.

mkinitcpio.conf

initcpio необходим для создания системы,способной загружаться с CD/DVD/USB.

Создайте mkinitcpio.conf:

```
$ vim mkinitcpio.conf
```

Обычно он содержит следующую информацию:

```
HOOKS="base udev archiso block usb fw filesystems usbinput"
```

Благодаря этому ваша система сможет загружаться с CD/DVD или USB. Стоит отметить, что автоопределение железа и прочее настраивается не здесь.

packages.list

Вам так-же понадобится список приложений, устанавливаемых на вашу live-систему. Как минимум вам понадобятся **base** и **kernel26**, но, вы вольны дополнять список приложениями на ваше усмотрение.

Примечание: mkarchiso использует */etc/pacman.conf* из основной системы. Если вы раскоментировали [testing], то приложения из него тоже будут использованы при создании образа.Если вы хотите использовать другой *pacman.conf*, вы можете создать его в папке проекта и использовать командой **mkarchiso -C pacman.conf**Для использования выделенного *pacman.conf*нужно добавить **-C pacman.conf** во все вызовы **mkarchiso** в Makefile.

```
$ vim packages.list
```

В список устанавливаемых пакетов будет разумно вставить, как минимум, следующее:

```
aufs2
aufs2-util
base
bash
coreutils
cpio
dhcpcd
dnsutils
file
fuse
kernel26
syslinux
nano
```

Этот список должен дать вам минимальную рабочую систему. Но не забывайте, что в ней не будет драйверов свыше включённых в ядро (видео, вай-фай, специализированные - их нужно будет добавить в список).

Tip: Вы так-же можете создать **custom local repository** для подготовки пакетов. Просто поставьте ваш локальный репозитарий на первую позицию в используемом **pacman.conf**.

isomounts

Вам понадобится файл, содержащий информацию о файловых системах, монтируемых при загрузке системы.

```
$ vim isomounts
```

Пример *isomounts*:

```
# archiso isomounts file
# img - location of image/directory to mount relative to addons directory
# arch - architecture of this image
# mount point - absolute location on the post-initrd root
# type - either 'bind' or 'squashfs' for now
# syntax: <img> <arch> <mount point> <type>
# ORDER MATTERS! Files take top-down precedence. Be careful
overlay.sqfs @ARCH@ / squashfs
root-image.sqfs @ARCH@ / squashfs
```

Важно: В конце файла должна быть пустая строка (EOF) иначе ждите, что система упадёт в kernel panic!

boot-files

Вам нужно будет добавить директорию "boot-files" и поддиректорию "isolinux/" содержащую "isolinux.cfg".

Важно: Ввиду последних изменений поддержка **GRUB** в Archiso была отменена. Пожалуйста, не используйте grub. Взамен, вы можете использовать isolinux из syslinux'a, при этом вы получите образ, который может быть записан не только на диск, но и на флэшку.

Использование isolinux

Использовать *isolinux* просто:

```
$ mkdir -p boot-files/isolinux/
$ vim boot-files/isolinux/isolinux.cfg
```

Образец:

```
prompt 1
timeout 0
display myarch.msg
DEFAULT myarch

LABEL myarch
KERNEL /boot/vmlinuz26
APPEND initrd=/boot/myarch.img archisolabel=XXX tmpfs_size=75%
locale=en_US.UTF-8

LABEL memtest86+
KERNEL /boot/memtest86+-2.10.bin
```

Возможно, вы захотите отображать сообщение над меню загрузки:

```
$ vim boot-files/isolinux/myarch.msg
```

Это может быть любое сообщение в ASCII:

```
HI GENTLEMEN LOL
WELCOME TO MY DISTRO
I HOPE U ENJOY MAKE UR TIME
HA-HA-HA
(ПРЕВЕД ДЖЕНТЕЛЬМЕНЫ Ы
 ДОБРО ПОЖАЛОВАТЬ В МОЙ ДИСТРИБУТИВЧЕГ
 НАДЕЮСЬ, ВАМ ПОНРАВИТСЯ ВРЕМЯ, КОТОРОЕ ВЫ НА НЕГО ПОТРАТИТЕ
 ГЫ-ГЫ-ГЫ)
```

Обратите внимание, что вам нужно будет где-нибудь достать memtest*.bin потому как в "поставку по умолчанию" он не входит. Если вы не хотите его использовать - закоментируйте.

Благодаря модульной структуре isolinux вы можете использовать большое колличество аддонов, потому что все *.c32 файлы скопированны и доступны вам. Подробней можете посмотреть **официальный сайт syslinux** и **archiso git-репозитарий**. Использование вышеперечисленных аддонов позволяет создавать красивые и сложные меню. Подробнее см. **тут**.

overlay

overlay предназначен для включения в дистрибутив бинарных репозитариев, конфигов, отличающихся от умолчальных и прочего. *mkarchiso* требует помещения всех файлов, предназначенных для оверлея, в одну директорию. Оверлей будет наложен на систему во время загрузки используя *AUFS*. Структура папки, содержащей файлы оверлея, должна повторять корневую систему.

Все файлы и директории, не существующие в оригинальной системе, но существующие в оверлее, будут созданы. Все файлы и директории существующие в оригинальной системе и в оверлее, будут перезаписаны оверлеем.

Создаём *overlay*:

```
$ mkdir overlay && cd overlay/
```

Это было легко, теперь надо наполнить оверлей полезняшками. Несколько примеров:

Примечание: Важно, чтобы всем файлам в оверлее были назначены правильные права доступа.Поэтому все изменения в директории оврелея рекомендуется производить из под рута

Добавление fstab

Если нужно добавить **fstab**:

```
$ mkdir etc
$ vim etc/fstab
aufs                   /             aufs      noauto              0      0
none                   /dev/pts      devpts    defaults            0      0
none                   /dev/shm      tmpfs     defaults            0      0
```

Добавление пользователя

Вручную

Так или иначе, но вам понадобятся пользователи в вашей live-системе. Есть много способов их добавления.Один из — скопировать файлы, для этого требуемые из корневой системы, и привести их в вид, удовлетворяющий вашим требованиям:

```
$ cp /etc/group etc/group
$ cp /etc/passwd etc/passwd
$ cp /etc/shadow etc/shadow
```

Примечание: Не оставляйте зашифрованный пароль passwd или shadow в файле! Пароль находится на второй позиции (после первого ':')

Пример беспарольного пользователя:

```
root::99999::::::
```

Так-же не забудьте создать домашнюю папку для пользователя (не забудьте изменить домашнюю папку в *passwd*). Для создания домашней папки во время загрузки и добавления туда /etc/skel пользовательской папки можно использовать rc.local. Если про /etc/skel в слышите впервые, вам, и правда, следует об этом прочитать.

```
$ vim etc/rc.local
mkdir /home/archie && chown archie:archie /home/archie
su -c "cp -r /etc/skel/.[a-zA-Z0-9]* /home/archie/" archie
```

Используя useradd

Другим способом добавления пользователя является использование *etc/rc.local* для создания пользователя при загрузке:

```
$ vim etc/rc.local
useradd -u 1000 -g users -G
storage,floppy,optical,audio,video,network,games,wheel,disk -d /home/archie
archie
Это создаст пользователя и домашнюю директорию для него.
```

Добавление чего-либо в домашнюю директорию пользователя при загрузке

Возможно, вы захотите добавить какие-то файлы кофигурации для пользователя live-системы.

Вам нужно будет создать следующую директорию, и поместить желаемое туда:

```
$ mkdir etc/skel
```

Для примера:

```
$ vim etc/skel/.bashrc
```

```
alias ls='ls --color=auto'
PS1='[\u@\h \W]\$ '
```

Описание всего, что можно сделать таким образом, значительно выходит за рамки данной статьи.

Примечание: Не пытайтесь использовать оверлей для прямого изменения /home/user/, это вызовет ошибки в правах доступа! Используйте /etc/skel/

Закрытие темы оверлея

Некоторые темы, не рассмотренные в данной статье (потому как рассмотрены в вики):

- Конфигурация *inittab* для старта X во время загрузки
- Конфигурация *hosts*
- Конфигурация *rc.conf*
- Конфигурация *sudoers*
- Конфигурация *rc.local*
- Добавление большего колличества полезняшек etc/skel
- Добавление большего колличества графического оформления
- Добавление разнообраных бинарников в opt/

Генерация образа

После всего времени, потраченного на конфигурацию, осталась самая приятная часть: Создание образа.

Это легко: Из под рута (это важно!) выполните следующую команду в директории вашего проекта (там, где лежит *Makefile*):

```
$ make all
```

На выходе вы получите .iso, готовый для записи на CD/DVD или USB:

```
dd if=my-image.iso of=/dev/some-usb-drive bs=8M
```

Примечание: Если на флеш носителе более одного раздела, не забудьте присвоить метку разделу, содержащему live-систему. Иначе с него будет невозможно загрузиться.

Важно: Осторожней с `dd`! Если вы используете его не на том устройстве, данные будет практически невозможно восстановить. Используйте с осторожностью.

Links

Archiso project page

Arch based distributions (active)

Categories:

- Live Arch systems (Русский)
- Русский