Archiso

Archiso is a highly-customizable tool for building Arch Linux live CD/USB ISO images. The official images are built with Archiso. It can be used as the basis for rescue systems, linux installers or other systems. This wiki article explains how to install Archiso, and how to configure it to control aspects of the resulting ISO image such as included packages and files. Technical requirements and build steps can be found in the official project documentation. Archiso is implemented with a number of bash scripts. The core component of Archiso is the *mkarchiso* command. Its options are documented in *mkarchiso -h* and not covered here.

Installation

Install the archiso or archiso-gitAUR package. The archiso-profilesAUR[broken link: package not found] package contains additional community-provided profiles.

Prepare a custom profile

Archiso comes with two profiles, **releng** and **baseline**.

- **releng** is used to create the official monthly installation ISO. It can be used as a starting point for creating a customized ISO image.
- **baseline** is a minimalistic configuration, that includes only the bare minimum packages required to boot the live environment from the medium.

To build an unmodified version of the profiles, skip to #Build the ISO. Otherwise, if you wish to adapt or customize one of archiso's shipped profiles, copy it from `/usr/share/archiso/configs/profile-name/` to a writable directory with a name of your choice. For example:

```
$ cp -r /usr/share/archiso/configs/releng/ archlive
```

Proceed to the following sections to customize and build the custom profile.

Profile structure

An archiso profile contains configuration that defines the resulting ISO image. The profile structure is documented in `/usr/share/doc/archiso/README.profile.rst`[1].

Selecting packages

Edit `packages.x86_64` to select which packages are to be installed on the live system image, listing packages line by line.

Custom local repository

To add packages not located in standard Arch repositories (e.g. custom packages or packages from AUR/ABS), set up a custom local repository and add your custom packages to it. Then add your repository to `pacman.conf` as follows:

```
archlive/pacman.conf
```

```
...
[customrepo]
SigLevel = Optional TrustAll
Server = file:///path/to/customrepo
...
```

Note:

- The ordering within `pacman.conf` matters. To give top priority to your custom repository, place it above the other repository entries.
- This `pacman.conf` is only used for building the image. It will not be used in the live environment. To do this, see #Adding repositories to the image.

Packages from multilib

To install packages from the multilib repository, simply uncomment that repository in `pacman.conf`.

Adding files to image

The airootfs directory is used as the starting point for the root directory (`/`) of the live system on the image. All its contents will be copied over to the working directory before packages are installed.

Place any custom files and/or directories in the desired location under `airootfs/`. For example, if you have a set of iptables scripts on your current system you want to be used on you live image, copy them over as such:

```
$ cp -r /etc/iptables archlive/airootfs/etc
```

Similarly, some care is required for special configuration files that reside somewhere down the hierarchy. Missing parts of the directory structure can be simply created with mkdir(1).

Tip: To add a file to the install user's home directory, place it in `archlive/airootfs/root/`. To add a file to all other users home directories, place it in `archlive/airootfs/etc/skel/`.
Note: Custom files that conflict with those provided by packages will be overwritten unless a package specifies them as backup files.

By default, permissions will be `644` for files and `755` for directories. All of them will be owned by the root user. To set different permissions or ownership for specific files and/or folders, use the `file_permissions` associative array in `profiledef.sh`. See README.profile.rst for details.

Adding repositories to the image

To add a repository that can be used in the live environment, create a suitably modified `pacman.conf` and place it in `archlive/airootfs/etc/`.

If the repository also uses a key, place the key in `archlive/airootfs/usr/share/pacman/keyrings/`. The key file name must end with `.gpg`. Additionally, the key must be trusted. This can be accomplished by creating a GnuPG exported trust file in the same directory. The file name must end with `-trusted`. The first field is the key fingerprint, and the second is the trust. You can reference `/usr/share/pacman/keyrings/archlinux-trusted` for an example.

archzfs example

The files in this example are:

```
airootfs
├── etc
│   ├── pacman.conf
│   └── pacman.d
│       └── archzfs_mirrorlist
└── usr
    └── share
        └── pacman
            └── keyrings
                ├── archzfs.gpg
                └── archzfs-trusted
```

```
airootfs/etc/pacman.conf
```

```
...
[archzfs]
Include = /etc/pacman.d/archzfs_mirrorlist
...
```

```
airootfs/etc/pacman.d/archzfs_mirrorlist
```

```
Server = https://archzfs.com/$repo/$arch
Server = https://mirror.sum7.eu/archlinux/archzfs/$repo/$arch
```

```
Server = https://mirror.biocrafting.net/archlinux/archzfs/$repo/$arch
Server = https://mirror.in.themindsmaze.com/archzfs/$repo/$arch
Server = https://zxcvfdsa.com/archzfs/$repo/$arch
```

```
airootfs/usr/share/pacman/keyrings/archzfs-trusted
```

```
DDF7DB817396A49B2A2723F7403BD972F75D9D76:4:
```

`archzfs.gpg` itself can be obtained directly from the repository site at https://archzfs.com/archzfs.gpg.

Kernel

Although both archiso's included profiles only have linux, ISOs can be made to include other or even multiple kernels.

First, edit `packages.x86_64` to include kernel package names that you want. When *mkarchiso* runs, it will include all *work_dir*`/airootfs/boot/vmlinuz-*` and *work_dir*`/boot/initramfs-*.img` files in the ISO (and additionally in the FAT image used for UEFI booting).

mkinitcpio presets by default will build fallback initramfs images. For an ISO, the main initramfs image would not typically include the `autodetect` hook, thus making an additional fallback image unnecessary. To prevent the creation of an fallback initramfs image, so that it does not take up space or slow down the build process, place a custom preset in *archlive*`/airootfs/etc/mkinitcpio.d/`*pkgbase*`.preset`. For example, for linux-lts:

archlive`/airootfs/etc/mkinitcpio.d/linux-lts.preset`

```
PRESETS=('archiso')

ALL_kver='/boot/vmlinuz-linux-lts'
ALL_config='/etc/mkinitcpio.conf'

archiso_image="/boot/initramfs-linux-lts.img"
```

Finally create boot loader configuration to allow booting the kernel(s).

Boot loader

Archiso supports syslinux for BIOS booting and GRUB or systemd-boot for UEFI booting. Refer to the articles of the boot loaders for information on their configuration syntax.

Tip:

- The **releng** profile by default builds into an ISO that supports both BIOS and UEFI booting when burned to an optical disc using El Torito, or when written to a hard disk (or USB flash drive, or similar) using Isohybrid.
- Due to the modular nature of isolinux, you are able to use lots of addons since all *.c32* files are copied and available to you. Take a look at the official syslinux site and the archiso git repo. Using said addons, it is possible to make visually attractive and complex menus. See [2].

mkarchiso expects that GRUB configuration is in the `grub` directory, systemd-boot configuration is in the `efiboot` directory, and syslinux configuration in `syslinux` and `isolinux` directories.

UEFI Secure Boot

If you want to make your Archiso bootable on a UEFI Secure Boot enabled environment, you must use a signed boot loader. You can follow the instructions on Secure Boot#Booting an installation medium.

systemd units

To enable systemd services/sockets/timers for the live environment, you need to manually create the symbolic links just as `systemctl enable` does it.

For example, to enable `gpm.service`, which contains `WantedBy=multi-user.target`, run:

```
$ mkdir -p archlive/airootfs/etc/systemd/system/multi-user.target.wants
$ ln -s /usr/lib/systemd/system/gpm.service archlive
/airootfs/etc/systemd/system/multi-user.target.wants/
```

The required symlinks can be found out by reading the systemd unit, or if you have the service installed, by enabling it and observing the systemctl output.

Login manager

Starting X at boot is done by enabling your login manager's systemd service. If you do not know which *.service* to enable, you can easily find out in case you are using the same program on the system you build your ISO on. Just use:

```
$ ls -l /etc/systemd/system/display-manager.service
```

Now create the same symlink in `archlive/airootfs/etc/systemd/system/`. For LXDM:

```
$ ln -s /usr/lib/systemd/system/lxdm.service archlive
/airootfs/etc/systemd/system/display-manager.service
```

This will enable LXDM at system start on your live system.

Changing automatic login

The configuration for getty's automatic login is located under `airootfs/etc/systemd/system/getty@tty1.service.d/autologin.conf`.

You can modify this file to change the auto login user:

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin username --noclear %I 38400 linux
```

Or remove `autologin.conf` altogether to disable auto login.

If you are using the serial console, create `airootfs/etc/systemd/system/serial-getty@ttyS0.service.d/autologin.conf` with the following content instead:

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty -o '-p -- \\u' --noclear --autologin root --keep-baud 115200,57600,38400
```

Users and passwords

To create a user which will be available in the live environment, you must manually edit `archlive/airootfs/etc/passwd`, `archlive/airootfs/etc/shadow`, `archlive/airootfs/etc/group` and `archlive/airootfs/etc/gshadow`.

Note: If these files exist, they must contain the root user and group.
For example, to add a user `archie`. Add them to `archlive/airootfs/etc/passwd` following the passwd(5) syntax:

`archlive/airootfs/etc/passwd`

```
root:x:0:0:root:/root:/usr/bin/zsh
archie:x:1000:1000::/home/archie:/usr/bin/zsh
```

Note: The `passwd` file must end with a newline.

Add the user to `archlive/airootfs/etc/shadow` following the syntax of shadow(5). If you want to define a password for the user, generate a password hash with `openssl passwd -6` and add it to the file. For example:

`archlive/airootfs/etc/shadow`

```
root::14871::::::
archie:$6$randomsalt$cij4/pJREFQV/NgAgh9YyBIoCRRNq2jp5l8lbnE5aLggJnzIRmNVlogAg8N6hEEecLwXHtMQIl2
```

Otherwise, you may keep the password field empty, meaning that the user can log in with no password.

Add the user's group and the groups which they will part of to `archlive/airootfs/etc/group` according to group(5). For example:

archlive/airootfs/etc/group

```
root:x:0:root
adm:x:4:archie
wheel:x:10:archie
uucp:x:14:archie
archie:x:1000:
```

Create the appropriate `archlive/airootfs/etc/gshadow` according to gshadow(5):

archlive/airootfs/etc/gshadow

```
root:!*::root
archie:!*::
```

Make sure `/etc/shadow` and `/etc/gshadow` have the correct permissions:

archlive/profiledef.sh

```
...
file_permissions=(
  ...
  ["/etc/shadow"]="0:0:0400"
  ["/etc/gshadow"]="0:0:0400"
)
```

After package installation, *mkarchiso* will create all specified home directories for users listed in `archlive/airootfs/etc/passwd` and copy `work_directory/x86_64/airootfs/etc/skel/*` to them. The copied files will have proper user and group ownership.

Changing the distribution name used in the ISO

Start by copying the file `/etc/os-release` into the `etc/` folder in the rootfs. Then, edit the file accordingly. You can also change the name inside of GRUB and syslinux.

Build the ISO

Build an ISO which you can then burn to CD or USB by running:

```
# mkarchiso -v -w /path/to/work_dir -o /path/to/out_dir /path/to/profile/
```

- `-w` specifies the working directory. If the option is not specified, it will default to `work` in the current directory.
- `-o` specifies the directory where the built ISO image will be placed. If the option is not specified, it will default to `out` in the current directory.
- It should be noted the profile file `profiledef.sh` cannot be specified when running mkarchiso, only the path to the file.

Replace `/path/to/profile/` with the path to your custom profile, or with `/usr/share/archiso/configs/releng/` if you are building an unmodified profile.

Tip: If memory allows, it is preferred to place the working directory on tmpfs. E.g.:

```
# mkarchiso -v -w /tmp/archiso-tmp /path/to/profile/
```

When run, the script will download and install the packages you specified to `work_directory/x86_64/airootfs`, create the kernel and init images, apply your customizations and finally build the ISO into the output directory.

Removal of work directory

Warning: If *mkarchiso* is interrupted, run findmnt(8) to make sure there are no mount binds before deleting it - otherwise, **you may lose data** (e.g. an external device mounted at `/run/media/user/label` gets bound within `work/x86_64/airootfs/run/media/user/label` during the build process).

The temporary files are copied into work directory. After successfully building the ISO , the work directory and its contents can be deleted. E.g.:

```
# rm -rf /path/to/work_dir
```

Using the ISO

See Installation guide#Prepare an installation medium for various options.

Test the ISO in QEMU

Install the optional dependencies qemu-desktop and edk2-ovmf.

Use the convenience script `run_archiso` to run a built image using QEMU.

```
$ run_archiso -i /path/to/archlinux-yyyy.mm.dd-x86_64.iso
```

The virtual machine can also be run using UEFI emulation:

```
$ run_archiso -u -i /path/to/archlinux-yyyy.mm.dd-x86_64.iso
```

Tips and tricks

Online build

If you do not have an arch system available or you need to setup Archiso from another GNU/Linux distribution, be aware there exists an online builder.

Prepare an ISO for an installation via SSH

Note: Since `archlinux-2021.02.01-x86_64.iso`, cloud-init support is provided, and `sshd.service` is enabled by default.

To install Arch Linux via SSH without any interaction with the system, an SSH public key must be placed in `authorized_keys`.

Adding the SSH key can either be done manually (explained here), or by cloud-init.

To add the key manually, first copy Archiso's releng profile to a writable directory. The following example uses `archlive`.

```
$ cp -r /usr/share/archiso/configs/profile/ archlive
```

Create a `.ssh` directory in the home directory of the user which will be used to log in. The following example will be using the root user.

```
$ mkdir archlive/airootfs/root/.ssh
```

Add the SSH public key(s), which will be used to log in, to `authorized_keys`:

```
$ cat ~/.ssh/key1.pub >> archlive/airootfs/root/.ssh/authorized_keys
$ cat ~/.ssh/key2.pub >> archlive/airootfs/root/.ssh/authorized_keys
```

Set correct permissions and ownership for the `.ssh` directory and the `authorized_keys` file:

```
archlive/profiledef.sh
```

```
...
file_permissions=(
  ...
  ["/root"]="0:0:0750"
  ["/root/.ssh"]="0:0:0700"
  ["/root/.ssh/authorized_keys"]="0:0:0600"
)
```

Finally build the ISO. Upon booting the ISO, OpenSSH will start and it will be possible to log in using the corresponding SSH private key(s).

Automatically connect to a Wi-Fi network using iwd

Create `/var/lib/iwd/` inside the profile's `airootfs` directory and set the correct permissions:

```
$ mkdir -p archlive/airootfs/var/lib/iwd
```

```
archlive/profiledef.sh
```

```
...
file_permissions=(
  ...
  ["/var/lib/iwd"]="0:0:0700"
)
```

Follow the instructions in iwd#Network configuration and iwd.network(5) to create a network configuration file for your Wi-Fi network.

Save the configuration file inside `archlive/airootfs/var/lib/iwd/`.

Adjusting the size of the root file system

When installing packages in the live environment, for example on hardware requiring DKMS modules, the default size of the root file system might not allow the download and installation of such packages due to its size.

Tip: See BBS#210389 for the reason behind the chosen size, and FS#45618 for historical details.

It will manifest as the following error message when downloading files or installing packages in the live environment:

```
error: partition / too full: 63256 blocks needed, 61450 blocks free
error: not enough free disk space
error: failed to commit transaction (not enough free disk space)
```

```
Errors occurred: no packages were upgraded.
```

To adjust the size on the fly:

```
# mount -o remount,size=SIZE /run/archiso/cowspace
```

See tmpfs(5) § size for the possible parameters of *SIZE*.

To adjust the size at the bootloader stage (by pressing `e` or `Tab`) use the boot option:

```
cow_spacesize=SIZE
```

To adjust the size while building an image add the boot option to:

- `efiboot/loader/entries/*.cfg`
- `grub/*.cfg`
- `syslinux/*.cfg`

The result can be checked with:

```
$ df -h
```

See mkinitcpio-archiso boot parameters.

Encryption

In order for vanilla `mkarchiso` to produce encrypted images, LUKS support in archiso, encrypt hook's compatibility in mkinitcpio-archiso and nested `cryptkey`s support in cryptsetup merge requests need to be approved.

Packages with such features already merged are archiso-encryptionAUR[broken link: package not found], mkinitcpio-archiso-encryptionAUR[broken link: package not found] and cryptsetup-nested-cryptkeyAUR[broken link: package not found].

To enable encryption on an existing profile:

- add `+luks` to the `airootfs_image_type` value in `profiledef.sh`;
- set an `encryption_key` in `profiledef.sh` (to use a key file instead of a password).
- enable the `encrypt` hook in `/etc/mkinitcpio.conf`;
- add AUR packages (or build custom replacements with the aforementioned sources) to `packages.x86_64`
- add the `keys` buildmode to the `buildmodes` array in `profiledef.sh` (to build a second ISO containing the key file that put on external storage is able to boot the system).

Example configurations based on the `baseline` and `releng` profiles are available as `ebaseline` and `ereleng` in the archiso-profilesAUR[broken link: package not found] package.

Google Compute Engine images

A Google Compute Engine-compatible `releng` compressed image is available as archlinux-gceAUR[broken link: package not found].

Libvirt VM configuration

A libvirt configuration which runs the `releng` image is available as archlinux-libvirtAUR[broken link: package not found].

Troubleshooting

Window manager freezes

If you want to use a window manager in the Live CD, you must add the necessary and correct video drivers, or the WM may freeze on loading.

See also

- Archiso project page
- Official documentation
- Arch Linux Release Engineering mailing list
- #archlinux-releng — Arch Linux Release Engineering IRC channel
- archiso-manager — the tool used for building the official monthly ISOs

---

Revision #1
Created 2024-04-30 04:55:13 UTC by Miles Menninga
Updated 2024-04-30 04:55:45 UTC by Miles Menninga