python-archinstall

Release v2.3.0

Anton Hvornum

Jan 14, 2022

RUNNING THE INSTALLER

archinstall is library which can be used to install Arch Linux.
The library comes packaged with different pre-configured installers, such as the default *Guided installation* installer.

A demo of the *Guided installation* installer can be seen here: https://www.youtube.com/watch?v=9Xt7X_Iqg6E.

Some of the features of Archinstall are:

- **No external dependencies or installation requirements.** Runs without any external requirements or installation processes.
- **Context friendly.** The library always executes calls in sequential order to ensure installation-steps don't overlap or execute in the wrong order. It also supports *(and uses)* context wrappers to ensure cleanup and final tasks such as `mkinitcpio` are called when needed.
- **Full transparancy** Logs and insights can be found at `/var/log/archinstall` both in the live ISO and the installed system.
- **Accessibility friendly** Archinstall works with `espeakup` and other accessibility tools thanks to the use of a TUI.

CHAPTER

ONE

GUIDED INSTALLATION

This is the default script the Arch Linux Archinstall package.
It will guide you through a very basic installation of Arch Linux.

Note: There are other scripts and they can be invoked by executing *archinstall <script> (without .py)*. To see a complete list of scripts, see the source code directory examples/

The installer has three pre-requisites:

- The latest version of Arch Linux ISO
- A physical or virtual machine to install on
- A working internet connection prior to running archinstall

Note: A basic understanding of machines, ISO-files and command line arguments are needed. Please read the official Arch Linux Wiki to learn more about your future operating system.

Warning: The installer will not configure WiFi before the installation begins. You need to read up on Arch Linux networking before you continue.

1.1 Running the guided installation

To start the installer, run the following in the latest Arch Linux ISO:

```
archinstall --script guided
```

The `--script guided` argument is optional as it's the default behavior.
But this will use our most guided installation and if you skip all the option steps it will install a minimal Arch Linux experience.

1.2 Installing directly from a configuration file

The guided installation also supports installing with pre-configured answers to all the guided steps.
This can be a quick and convenient way to re-run one or several installations.

After each successful installation a pre-configured configuration will be found at `/var/log/archinstall` both on the live media and the installed system.

There are three different configuration files, all of which are optional.

- `--config` that deals with the general configuration of language and which profiles to use.
- `--creds` which takes any `superuser`, `user` or `root` account data.
- `--disk_layouts` for defining the desired partition strategy on the selected `"harddrives"` in `--config`.

Note: You can always get the latest options with `archinstall --dry-run`, but edit the following json according to your needs. Save the configuration as a `.json` file. Archinstall can source it via a local or remote path (URL)

```
{
    "audio": "pipewire",
    "bootloader": "systemd-bootctl",
    "custom-commands": [
        "cd /home/devel; git clone https://aur.archlinux.org/paru.git",
        "chown -R devel:devel /home/devel/paru",
        "usermod -aG docker devel"
    ],
    "filesystem": "btrfs",
    "gfx_driver": "VMware / VirtualBox (open-source)",
    "harddrives": [
        "/dev/nvme0n1"
    ],
    "swap": true,
    "hostname": "development-box",
    "kernels": [
        "linux"
    ],
    "keyboard-language": "us",
    "mirror-region": "Worldwide",
    "nic": {
        "NetworkManager": true,
        "nic": "Use NetworkManager (necessary to configure internet graphically in GNOME␣
↪and KDE)"
    },
    "ntp": true,
    "packages": ["docker", "git", "wget", "zsh"],
    "profile": "gnome",
    "services": ["docker"],
    "sys-encoding": "utf-8",
    "sys-language": "en_US",
    "timezone": "US/Eastern",
}
```

To use it, assuming you put it on `https://domain.lan/config.json`:

```
archinstall --config https://domain.lan/config.json
```

1.3 Options for `--config`

(To see which keys are required, scroll to the right in the above table.)

Key	Values	Description	Required
audio	pipewire/pulseaudio	Audioserver to be installed	No
bootloader	systemd-bootctl/grub-install	Bootloader to be installed *(grub being mandatory on BIOS machines)*	Yes
custom-commands	[<command1>, <command2>, . . .]	Custom commands to be run post install	No
gfx_driver	• "VMware / VirtualBox (open-source)" • "Nvidia" • "Intel (open-source)" • "AMD / ATI (open-source)" • "All open-source (default)"	Graphics Drivers to install	No
harddrives	[<path of device>, <path of second device>, . . . }	Multiple paths to block devices to be formatted	No[1]
hostname	any	Hostname of machine after installation. Default will be `archinstall`	No
kernels	["kernel1", "kernel2"]	List of kernels to install eg: linux, linux-lts, linux-zen etc	Atleast 1
keyboard-language	Any valid layout given by `localectl list-keymaps`	eg: `us`, `de` or `de-latin1` etc. Defaults to `us`	No
mirror-region	{"<Region Name>": { "Mirror URL": True/False}, ..} "Worldwide" or "Sweden"	Defaults to automatic selection. Either takes a dictionary structure of region and a given set of mirrors. Or just a region and archinstall will source any mirrors for that region automatically	No
nic	{ NetworkManager: <boolean> } { "eth0": {"address": "<ip>", "subnet": "255.0.0.0"}} "Copy ISO network configuration to installation"	Takes three different kind of options. Copy, NetworkManager or a nic name. Copy will copy the network configuration used in the live ISO. NetworkManager will install and configure NetworkManager	No
ntp	<boolean>	Set to true to set-up ntp post install	No
packages	["package1", "package2", ..]	List of packages to install post-installation	No
profile	Name of the profile to in-	Profiles are present in pro-	No

Note: [1] If no entires are found in `harddrives`, archinstall guided installation will use whatever is mounted currently under `/mnt/archinstall`.

1.4 Options for `--creds`

Creds is a separate configuration file to separate normal options from more sensitive data like passwords.
Below is an example of how to set the root password and below that are description of other values that can be set.

```
{
    "!root-password" : "SecretSanta2022"
}
```

Key	Values	Description	Required
!encryption-password	any	Password to encrypt disk, not encrypted if password not provided	No
!root-password	any	The root account password	No
!superusers	{ "<username>": { "!password": "<password>"}, ..}	List of superuser credentials, see configuration for reference	Yes[1]
!users	{ "<username>": { "!password": "<password>"}, ..}	List of regular user credentials, see configuration for reference	No

Note: [1] `!superusers` is optional only if `!root-password` was set. `!superusers` will be enforced otherwise and the minimum amount of superusers required will be set to 1.

1.5 Options for `--disk_layouts`

Note:

The layout of `--disk_layouts` is a bit complicated.
It's highly recommended that you generate it using `--dry-run` which will simulate an installation, without performing any damaging actions on your machine. *(no formatting is done)*

```
{
    "/dev/loop0": {
        "partitions": [
            {
                "boot": true,
                "encrypted": false,
                "filesystem": {
                    "format": "fat32"
                },
```

(continues on next page)

(continued from previous page)

```
                "format": true,
                "mountpoint": "/boot",
                "size": "513MB",
                "start": "5MB",
                "type": "primary"
            },
            {
                "btrfs": {
                    "subvolumes": {
                        "@.snapshots": "/.snapshots",
                        "@home": "/home",
                        "@log": "/var/log",
                        "@pkgs": "/var/cache/pacman/pkg"
                    }
                },
                "encrypted": true,
                "filesystem": {
                    "format": "btrfs"
                },
                "format": true,
                "mountpoint": "/",
                "size": "100%",
                "start": "518MB",
                "type": "primary"
            }
        ],
        "wipe": true
    }
}
```

The overall structure is that of `{ "blockdevice-path" : ...}` followed by options for that blockdevice. Each partition has it's own settings, and the formatting is executed in order *(top to bottom in the above example)*. Mountpoints is later mounted in order of path traversal, `/` before `/home` etc.

Key	Values	Description	Required
filesystem	{ "format": "ext4 / btrfs / fat32 etc." }	Filesystem for root and other partitions	Yes
boot	<bool>	Marks the partition as bootable	No
encrypted	<bool>	Mark the partition for encryption	No
mountpoint	/path	Relative to the inside of the installation, where should the partition be mounted	Yes
start	<size><B, MiB, GiB, %, etc>	The start position of the partition	Yes
type	primary	Only used if MBR and BIOS is used. Marks what kind of partition it is.	No
btrfs	{ "subvolumes": {"subvolume": "mountpoint"}}	Support for btrfs subvolumes for a given partition	No

CHAPTER

TWO

DISCORD

There's a discord channel which is frequented by some contributors.

To join the server, head over to https://discord.gg/cqXU88y and join in.

There's not many rules other than common sense and to treat others with respect. The general chat is for off-topic things as well.

There's the `@Party Animals` role if you want notifications of new releases which is posted in the `#Release Party` channel. Another thing is the `@Contributors` role can be activated by contributors by writing `!verify` and follow the verification process.

Hop in, we hope to see you there! :)

CHAPTER

THREE

ISSUE TRACKER & BUGS

Issues and bugs should be reported over at https://github.com/archlinux/archinstall/issues.

General questions, enhancements and security issues can be reported over there too. For quick issues or if you need help, head over to the Discord server which has a help channel.

3.1 Log files

When submitting a help ticket, please include the `/var/log/archinstall/install.log`.
It can be found both on the live ISO but also in the installed filesystem if the base packages were strapped in.

Tip:

An easy way to submit logs is `curl -F'file=@/var/log/archinstall/install.log' https://0x0.st`.
Use caution when submitting other log files, but `archinstall` pledges to keep `install.log` safe for posting publicly!

There are additional log files under `/var/log/archinstall/` that can be useful:

- `/var/log/archinstall/user_configuration.json` - Stores most of the guided answers in the installer
- `/var/log/archinstall/user_credentials.json` - Stores any usernames or passwords, can be passed to `--creds`
- `/var/log/archinstall/user_disk_layouts.json` - Stores the chosen disks and their layouts
- `/var/log/archinstall/install.log` - A log file over what steps were taken by archinstall
- `/var/log/archinstall/cmd_history.txt` - A complete command history, command by command in order
- `/var/log/archinstall/cmd_output.txt` - A raw output from all the commands that were executed by archinstall

Warning: We only try to guarantee that `/var/log/archinstall/install.log` is free from sensitive information. Any other log file should be pasted with **utmost care**!

CHAPTER

FOUR

PYTHON LIBRARY

Archinstall ships on PyPi as archinstall. But the library can be installed manually as well.

> **Warning:** These steps are not required if you want to use archinstall on the official Arch Linux ISO.

4.1 Installing with pacman

Archinstall is on the official repositories. And it will also install archinstall as a python library.

To install both the library and the archinstall script:

```
pacman -S archinstall
```

Alternatively, you can install only the library and not the helper executable using the `python-archinstall` package.

4.2 Installing with PyPi

The basic concept of PyPi applies using *pip*.

```
pip install archinstall
```

4.3 Install using source code

You can also install using the source code.
For sake of simplicity we will use `git clone` in this example.

```
git clone https://github.com/archlinux/archinstall
```

You can either move the folder into your project and simply do

```
import archinstall
```

Or you can use setuptools to install it into the module path.

```
sudo python setup.py install
```

CHAPTER

FIVE

PYTHON MODULE

Archinstall supports running in module mode. The way the library is invoked in module mode is limited to executing scripts under the **example** folder.

It's therefore important to place any script or profile you wish to invoke in the examples folder prior to building and installing.

5.1 Pre-requisites

We'll assume you've followed the *Install using source code* method. Before actually installing the library, you will need to place your custom installer-scripts under *./archinstall/examples/* as a python file.

More on how you create these in the next section.

> **Warning:** This is subject to change in the future as this method is currently a bit stiff. The script path will become a parameter. But for now, this is by design.

5.2 Creating a script

Lets create a *test_installer* - installer as an example. This is assuming that the folder *./archinstall* is a git-clone of the main repo. We begin by creating *./archinstall/examples/test_installer.py*. The placement here is important later.

This script can now already be called using *python -m archinstall test_installer* after a successful installation of the library itself. But the script won't do much. So we'll do something simple like list all the hard drives as an example.

To do this, we'll begin by importing *archinstall* in our *./archinstall/examples/test_installer.py* and call some functions.

```
import archinstall

all_drives = archinstall.list_drives()
print(all_drives)
```

This should print out a list of drives and some meta-information about them. As an example, this will do just fine.

Now, go ahead and install the library either as a user-module or system-wide.

5.3 Calling a module

Assuming you've followed the example in *Creating a script*, you can now safely call it with:

```
python -m archinstall test_installer
```

This should now print all available drives on your system.

Note: This should work on any system, not just Arch Linux based ones. But note that other functions in the library rely heavily on Arch Linux based commands to execute the installation steps. Such as *arch-chroot*.

CHAPTER

SIX

BINARY EXECUTABLE

Archinstall can be compiled into a standalone executable. For Arch Linux based systems, there's a package for this called archinstall.

Warning: This is not required if you're running archinstall on a pre-built ISO. The installation is only required if you're creating your own scripted installations.

6.1 Using pacman

Archinstall is on the official repositories.

```
sudo pacman -S archinstall
```

6.2 Using PKGBUILD

The source contains a binary PKGBUILD which can be either copied straight off the website or cloned using `git clone https://github.com/Torxed/archinstall`.

Once you've obtained the *PKGBUILD*, building it is pretty straight forward.

```
makepkg -s
```

Which should produce an *archinstall-X.x.z-1.pkg.tar.zst* which can be installed using:

```
sudo pacman -U archinstall-X.x.z-1.pkg.tar.zst
```

Note: For a complete guide on the build process, please consult the PKGBUILD on ArchWiki.

6.3 Manual compilation

You can compile the source manually without using a custom mirror or the *PKGBUILD* that is shipped. Simply clone or download the source, and while standing in the cloned folder *./archinstall*, execute:

```
nuitka3  --standalone --show-progress archinstall
```

This requires the nuitka package as well as *python3* to be installed locally.

CHAPTER

SEVEN

BINARY EXECUTABLE

Warning: The binary option is limited and stiff. It's hard to modify or create your own installer-scripts this way unless you compile the source manually. If your usecase needs custom scripts, either use the pypi setup method or you'll need to adjust the PKGBUILD prior to building the arch package.

The binary executable is a standalone compiled version of the library. It's compiled using nuitka with the flag *–standalone*.

7.1 Executing the binary

As an example we'll use the guided installer. To run the *guided* installed, all you have to do *(after installing or compiling the binary)*, is run:

```
./archinstall guided
```

As mentioned, the binary is a bit rudimentary and only supports executing whatever is found directly under *./archinstall/examples*. Anything else won't be found. This is subject to change in the future to make it a bit more flexible.

CHAPTER

EIGHT

ARCHINSTALL.INSTALLER

The installer is the main class for accessing an installation-instance. You can look at this class as the installation you have or will perform.

Anything related to **inside** the installation, will be found in this class.

CHAPTER

NINE

ARCHINSTALL.PROFILE

This class enables access to pre-programmed profiles. This is not to be confused with *archinstall.Application* which is for pre-programmed application profiles.

Profiles in general is a set or group of installation steps. Where as applications are a specific set of instructions for a very specific application.

An example would be the *(currently fictional)* profile called *database*. The profile *database* might contain the application profile *postgresql*. And that's the difference between *archinstall.Profile* and *archinstall.Application*.

CHAPTER

TEN

ARCHINSTALL.APPLICATION

This class enables access to pre-programmed application configurations. This is not to be confused with *archinstall.Profile* which is for pre-programmed profiles for a wider set of installation sets.

Warning: All these helper functions are mostly, if not all, related to outside-installation-instructions. Meaning the calls will affect your current running system - and not touch your installed system.

CHAPTER

ELEVEN

PROFILE RELATED HELPERS

CHAPTER

TWELVE

PACKAGES

CHAPTER

THIRTEEN

LOCALE RELATED

CHAPTER

FOURTEEN

SERVICES

CHAPTER

FIFTEEN

MIRRORS

CHAPTER

SIXTEEN

DISK RELATED

CHAPTER

SEVENTEEN

LUKS (DISK ENCRYPTION)

CHAPTER

EIGHTEEN

NETWORKING

CHAPTER

NINETEEN

GENERAL

CHAPTER

TWENTY

EXCEPTIONS