Basic Arch Linux Installation Guide

by @uditkarode

August 3, 2020

This document guides you through a very basic installation of *Arch Linux.*

1 Preparations

This guide assumes you have a UEFI-based machine with a GPT partition table (all the new machines confirm to this), and that you already have a USB stick ready (with the ISO obtained from `https://www.archlinux.org/download/`), and have booted into it, hence obtaining a shell. This is where you're supposed to enter commands.

1.1 Deciding install partition

You first need to decide what partition you're going to install Arch on. You can examine the current partition table using the command `fdisk -l`, and make adjustments or create new partitions using the `cfdisk` command. An example partition table that `fdisk -l` would output is:

```
Device Start End Sectors Size Type
/dev/nvme0n1p1 2048 1333247 1331200 650M EFI System
/dev/nvme0n1p2 1333248 211048447 209715200 100G Linux filesystem
/dev/nvme0n1p4 420763648 791891967 371128320 177G Apple APFS
```

In my case, `/dev/nvme0n1p2` would be my *target partition.* We'll refer to the target partition as `<target>` in the rest of this document.

1.2 Readying partition for install

Once you've decided the *target partition*, you need to run the `mkfs.ext4` command on it this way:

```
mkfs.ext4 /dev/nvme0n1p2
```

This command will format your disk and set up an empty `ext4` filesystem on it.

Once the execution of this command finishes, you need to *mount* the drive. Mounting basically refers to attaching the desired filesystem to a directory on the current filesystem. We'll mount it at the `/mnt` directory this way:

```
mount /dev/<target> /mnt
```

You now have your partition ready and mounted, that ends the first section!

2 Connecting to the internet

2.1 Activating the connection

From now on, we're going to install packages from the Arch repositories, which requires us to be connected to the internet. **If you want to use Ethernet, just plug in the Ethernet cable and go straight to 2.2!**
Here's how you do it if you want to use WiFi:

```
wifi-menu
```

However, newer installation mediums for Arch Linux no longer contain the `netctl` package, so if you see `command not found` after executing `wifi-menu`, this is what you need to do:

- `iwctl`
  This will open the `iwctl` prompt.
- `device list`
  This command will output the WiFi devices that `iwctl` detects on your machine. Note the `name` of the device you want to use. It'll usually be `wlan0`. We'll refer to it as `<device>` now.
- `station <device> get-networks`
  This command will output the WiFi networks it detects around you. Note the network you want to connect to. We'll refer to it as `<wifiname>` from now on.
- `station <device> connect <wifiname>`
  If you don't see an error like `Operation failed`, you're good to go! Press `CTRL + D` to exit the `iwctl` prompt and return your shell.

2.2 Testing the connection

To test your connection to the internet, execute:

```
ping 8.8.8.8
```

You should see an output like:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=53.5 ms
```

Press `CTRL + C` to stop it.

If you see `connect: Network is unreachable`, something went wrong. You should retry section 2.1, or you won't be able to proceed.

3 Installation

3.1 Installing the `base`

We'll start off the installation process by installing the `base` and `base-devel` metapackages. the `base` metapackage contains all the basic packages that a system requires to function, while `base-devel` contains build tools and development related packages (even if you're not much of a developer, you might want to install this since it'll be used for AUR installs, more on that later). Here's the actual command:

```
pacstrap -i /mnt base base-devel
```

3.2 Fstab generation

Now we're going to need to generate the fstab file, which tells your system what partitions it needs to mount and their mountpoints. It's basically telling Linux what partition everything's installed in. To generate it, you can use the following command:

```
genfstab -U -p /mnt > /mnt/etc/fstab
```

3.3 Chrooting into our installation

At this point, you have a skeleton set up with all the basic packages in your target partition. However, we're missing some important packages, such as *the kernel itself.* That's what we're going to install in this subsection. Let's first get into the system!

```
arch-chroot /mnt
```

> But you said we don't have the kernel yet, how are we inside our installed system?

You aren't completely inside your new installation yet! What the `chroot` command does is that it changes the apparent root directory (`/`) for your currently

booted session, hence creating a sort of a sandbox. It's still using the kernel that came with the Arch installation drive.

3.4 Installation of necessary packages

3.4.1 Boot related packages

```
pacman -S grub os-prober efibootmgr
```

where:
`grub` is our bootloader
`os-prober` is used to find other EFI operating systems `efibootmgr` is a helper tool for GRUB installation tool

3.4.2 Kernel and firmware

```
pacman -S linux linux-headers linux-firmware
```

You could also install `linux-lts` and `linux-lts-headers` if you want the LTS kernel, but the cutting-edge Arch kernel is stable enough, so it's on you.

3.4.3 General use packages

```
pacman -S networkmanager sudo vim nano git zip unzip dialog
```

This installs packages that help connect to WiFi/Ethernet, and so on.

3.4.4 Processor/GPU Specific drivers

If you have an Intel CPU, install `intel-ucode` and if you have an AMD CPU, install `amd-ucode`. These packages ensure system stability.

Now for the **graphic drivers**, you have two choices - proprietary, or open-source drivers. Open-source drivers will usually do the job for you, so that's what I'm going to mention here.

NOTE: If you're using a laptop that has two GPUs (i.e. *Intel HD Graphics* **or** *AMD Radeon + Nvidia*), install the graphic drivers for the first/weaker GPU (may be *Intel HD Graphics* or *AMD Radeon*). This is because the more powerful GPU provided (usually *Nvidia*) is only supposed to be used for graphic intensive tasks, since the weaker GPU can usually handle all other tasks without any issues whatsoever. You'll need additional setup to be able to use the other/powerful GPU, and the configuration for it is out of scope for this guide.

- for **Nvidia**: `pacman -S mesa xf86-video-nouveau`

- for **Intel HD Graphics**: `pacman -S mesa xf86-video-intel`
- for **AMD Radeon**: `pacman -S mesa xf86-video-amdgpu`

In case you have a newer **Nvidia** GPU, you might want to install the proprietary drivers instead, since they perform a bit better. However, this is only needed if you plan to perform graphic-intensive tasks on this machine.

```
pacman -S dkms nvidia nvidia-utils nvidia-dkms nvidia-settings
```

That concludes the driver installation segment of the guide! Now you have an Arch Linux base with proper drivers for your system. You're getting closer to the end!

4 Configuration

4.1 Setting up your user account

Every person that uses a linux machine is supposed to have their own accounts. Let's make one for you - the primary user!

```
useradd -m -G wheel -s /bin/bash yournamehere
```

NOTE: make sure your username starts with a lowercase letter!

4.2 Letting you use sudo

`sudo` is a command that lets you execute other commands as the superuser. In order to use it, we must whitelist the group our user is in (`wheel`) in the sudo configs! To do this, execute:

```
echo '%wheel ALL=(ALL) ALL' >> /etc/sudoers
```

4.3 Setting passwords

To change the password for the root user, type in:

```
passwd
```

To change the password for your user, type in:

```
passwd yournamehere
```

Make sure to change the password for both the root user and your personal user. If you've done that, great work! This concludes the configuration section of the guide!

5 Setting up visual factors

Here, we set up how our installation looks. If you're coming from Windows or macOS, this might seem a bit strange to you. However, this is a common thing in the Linux world. What we did till now is just going to give us a command line interface. We now have choice over the Desktop Environment (commonly abbreviated as DE) we want to install. In this guide, we're going to go with Gnome, since this is what Ubuntu uses, and is easy to use and install for even new users. Here's what you need to do:

```
pacman -S gnome
```

That's it! This might install a lot of packages and take a lot of time.

6 Installing the bootloader

Now we're going to install the bootloader, `GRUB` onto our system.
Before we do this, you must find out the EFI partition of your system.
Again, we do this with `fdisk -l`. Let's look at a sample output.

```
Device Start End Sectors Size Type
/dev/nvme0n1p1 2048 1333247 1331200 650M EFI System
/dev/nvme0n1p2 1333248 211048447 209715200 100G Linux filesystem
/dev/nvme0n1p4 420763648 791891967 371128320 177G Apple APFS
```

In my case, `/dev/nvme0n1p1` is the EFI partition. We'll call this *efitarget* from now on. To install `GRUB`, run the following:

```
cd /
mkdir efidir
mount efitarget efidir
grub-install --target=x86_64-efi --efi-directory=esp --bootloader-id=GRUB
grub-mkconfig -o /boot/grub/grub.cfg
```

That does it!

7 Enabling required services

This is the last step of this guide. We need to enable `gdm` (Gnome Display Manager) and `NetworkManager` (the tool to let us use the internet), and set a proper locale. This is how you do it:

```
systemctl enable gdm
systemctl enable NetworkManager
```

```
localectl set-locale LANG=en_US.UTF-8
```

aaaand you're done!

8 Exiting and Rebooting

You're in the end game now :) all you need to do is:

```
exit
umount -R /mnt
reboot
```

Once that's done, remove the USB Stick, and wait for the computer to boot. It should boot right into GRUB. If not, you can change the boot order from your BIOS to make it boot GRUB. Once your system boots, you'll see the login screen. Enter the password you entered in section 3.3 and you should be in your system :)