Build Your Arch Linux System From Scratch

Arch Linux is a do-it-yourself Linux distro, It's very popular among Linux geeks and developers that like to really get at the nuts and bolts of a system. Arch give you the freedom to make any choice about the system. **It does not come with any pre-installed packages/drivers or graphical installer**, instead It uses a **command line installer**.
When you boot it up for the first time, you'll be greeted with a command-line tool. It expects you to perform the entire installation from the command-line and install all the necessary program/driver by yourself and customize it the way you want it — by piecing together the components that you'd like to include on your system.

Arch Linux is a really good way to learn what's going on inside a Linux box. You can learn a lot just from the installation process. I am going to walk through the base install, as well as several common post-install things like setting up networking, sound, mounts, X11 and video drivers, and adding users. I am not going to go in great detail on each step, so if you don't know how to do a certain step you may need to seek references elsewhere.

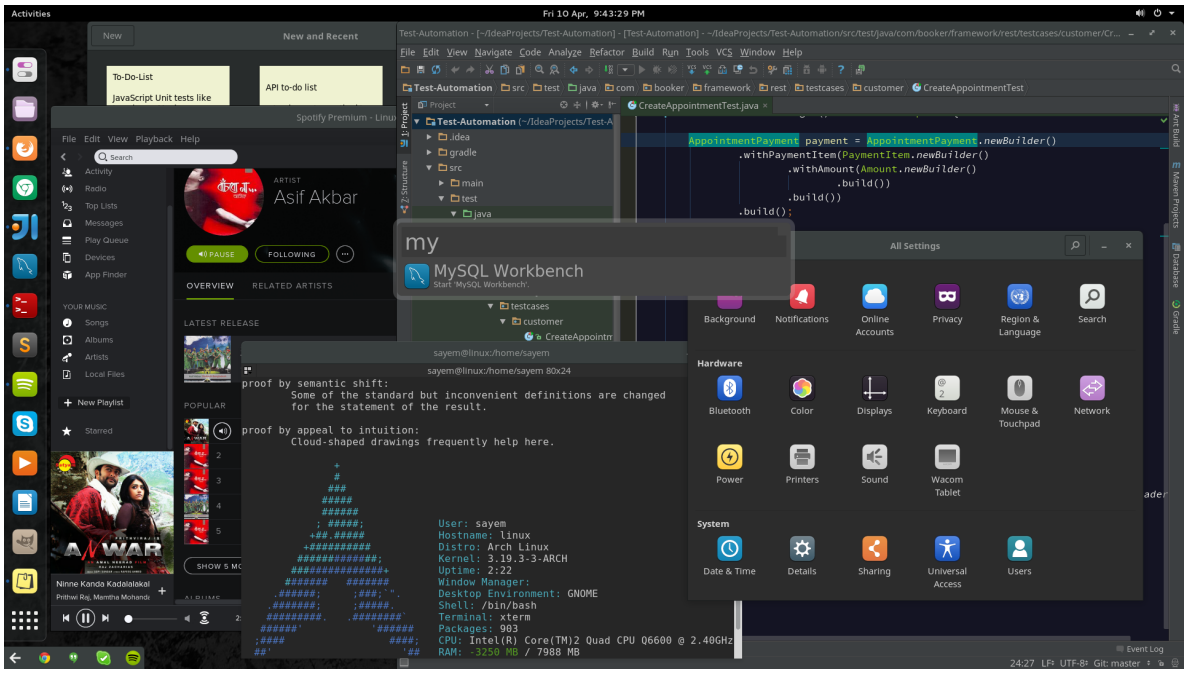
I'll also show you some tips, tricks and tweaks on how you can change the way the GNOME desktop looks and feel to suit your own personal tastes, that is, take a plain-vanilla GNOME Shell and transform it into a desktop that you like.

WARNING: There is a very **HIGH** chance you can destroy other operating systems or partition, if you don't do it right. **Please proceed with caution**. If you are new to Linux world I HIGHLY suggest you start off with a distro like Ubuntu or Mint Linux. Ubuntu is designed for people who want an off-the-shelf type system, where all of the choices are already made and the users are expected to sacrifice control for convenience.

Table of Contents

top

Here's a screenshot of my desktop, built just the way I want it



top

Bootable USB Installer

Download the ISO

First of all, you need the Arch Linux image, that can be downloaded from the Official Website.

> Arch Linux requires a x86_64 (i.e. 64 bit) compatible machine with a minimum of 512 MB RAM and 800 MB disk space for a minimal installation. However, it is recommended to have 2 GB of RAM and at least 20 GB of storage for a GUI to work without hassle.

After that, you should create the bootable flash drive with the Arch Linux image.

Linux

If you're on a Linux distribution, you can use the `dd` command for it. Like:

```
$ dd bs=4M if=/path/to/archlinux.iso of=/dev/sdx status=progress oflag=sync && sync
```

> Note that you need to update the `of=/dev/sdx` with your USB device location (it can be discovered with the `lsblk` command).

Otherwise, if you're on Windows, you can follow this tutorial.

MacOS

First, you need to identify the USB device. Open `/Applications/Utilities/Terminal` and list all storage devices with the command:

```
$ diskutil list
```

Your USB device will appear as something like `/dev/disk4 (external, physical)`. Verify that this is the device you want to erase by checking its name and size and then use its identifier for the commands below instead of /dev/disk4.

A USB device is normally auto-mounted in macOS, and you have to unmount (not eject) it before block-writing to it with `dd`. In Terminal, do:

```
$ diskutil unmountDisk /dev/disk4
```

Now copy the ISO image file to the device. The `dd` command is similar to its Linux counterpart, but notice the 'r' before 'disk' for raw mode which makes the transfer much faster:

```
# sudo dd if=/Users/sayem/Downloads/archlinux-2019.04.01-x86_64.iso  of=/dev/rdisk4
bs=1m
```

This command will run silently. To view progress, send SIGINFO by pressing `Ctrl+t`. Note `diskX` here should not include the `s1` suffix, or else the USB device will only be bootable in UEFI mode and not legacy. After completion, macOS may complain that "The disk you inserted was not readable by this computer". Select 'Ignore'. The USB device will be bootable.

top

BIOS

We'll install Arch Linux on UEFI mode, so you should check your bios settings. Boot into your hardware settings or BIOS or UEFI settings. Then check following settings. It can have different names and different keyboard shortcuts to reach it.

- Disable `Secure Boot`
- Disable `Launch CSM` or `Legacy Support`
- Set `Boot Mode` to `UEFI`
- Enable `USB Boot`

- Set USB Disk as boot priority

top

Pre-installation

Set Keyboard Layout

Arch Linux standard boots into an US keyboard layout. Many of us do not have to do anything. Just check the main keyboard keys and see if they all work.

Check boot mode

To check if the UEFI mode is enabled, run:

```
# ls /sys/firmware/efi/efivars
```

If the directory does not exists, the system may be booted in BIOS.

Internet Connection

If you are connected via Ethernet, you can test the connectivity by pinging `google`:

```
ping www.google.com -c 3
```

```
PING www.google.com (172.217.12.164) 56(84) bytes of data.
64 bytes from lga25s62-in-f4.1e100.net (172.217.12.164): icmp_seq=1 ttl=56 time=5.26 ms
64 bytes from lga25s62-in-f4.1e100.net (172.217.12.164): icmp_seq=2 ttl=56 time=7.86 ms
64 bytes from lga25s62-in-f4.1e100.net (172.217.12.164): icmp_seq=3 ttl=56 time=10.4 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6ms
rtt min/avg/max/mdev = 5.256/7.841/10.405/2.104 ms
```

Configure the system time

Once connected to the internet, it is important to synchronize the system time. This can be done by turning on the **Network Time Protocol (NTP)**. We will first check is the service is working and if not, we will activate it.

You can check the NTP service as below:

```
sh-4.3# timedatectl status
```

If the service is not working, you can set up it as below

```
sh-4.3# timedatectl set-ntp true
```

Partitioning

If you have multiple hard-drives in your machine, please unplug it now so you don't end up formatting them by mistake. Plug only the one that you will format to install Arch Linux on it.

I have 1TB of storage with 8 partitions, described on the following table below: (I will create two partitions `/dev/sdc5` and `/dev/sdc6` to install Arch Linux)

Partition	Name	File System	Size
/dev/sdc1	EFI System Partition	fat32	200M
/dev/sdc2	Macintosh HD	apfs	558G
/dev/sdc3	Ubuntu	ext4	70G
/dev/sdc4	Linux Swap	linux-swap	2G
/dev/sdc5	Arch Linux	ext4	67G
/dev/sdc6	Microsoft Reserved Partition	unknown	16M
/dev/sdc7	Windows 10	ntfs	232G
/dev/sdc8	Windows Recovery Environment	ntfs	481M

I'll use `cfdisk` to create `sdc4` and `sdc5` partitions:

```
root@arch ~ # cfdisk /dev/sdc
```

1. First, select the option **New** in the **cfdisk** menu.
2. Now specify the size of the partition you want to create. In my example:
    - `sdc4` - allocate **2G** for swap memory
    - `sdc5` - allocate **67G** for `/`
3. To save the changes, select the option **write** in the **cfdisk** menu.
4. Print the partition table and verify the new partition using `fdisk -l /dev/sdc` command.

Now that our partitions have been created, we can format it as below:

```
root@arch ~ # mkfs.ext4  /dev/sdc5
```

Now we should mount the partitions which have been created and formatted so that Arch Linux can point to them. We will mount the root partition to the `/mnt` folder.

```
root@arch ~ # mount /dev/sdc5 /mnt
```

The process for swap partition is slight different:

```
root@arch ~ # mkswap /dev/sdc4
root@arch ~ # swapon /dev/sdc4
```

Enabling a specific mirror

Before installation, is recommended to select the best mirror servers. So open the file `/etc/pacman.d/mirrorlist` (using `nano` or `vi` to do that) and move the best mirror to the top of the file.

```
root@arch ~ # nano /etc/pacman.d/mirrorlist
```

```
Server = https://arch.mirror.constant.com/$repo/os/$arch
Server = https://mirror.wdc1.us.leaseweb.net/archlinux/$repo/os/$arch
Server = https://mirrors.rit.edu/archlinux/$repo/os/$arch
Server = https://archlinux.olanfa.rocks/$repo/os/$arch
Server = https://arch.mirror.square-r00t.net/$repo/os/$arch
```

> **Tip**: That link generates a mirror list based on your location, you can use them as reference.

top

Installation

Now that the mirrors are already set, we can use `pacstrap` to install the Arch Linux bases system with the command below:

```
root@arch ~ # pacstrap /mnt base base-devel
```

Generate fstab

Generate a fstab with the `genfstab` script to define how disk partitions, block devices or remote file systems are mounted into the filesystem.

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

Chroot

We can use the `chroot` command to access your newly-installed operating system.

```
root@arch ~ # arch-chroot /mnt
```

> Now, if you want to install some package, do it with `pacman -S <package_name>`

Check pacman keys

```
# pacman-key --init
# pacman-key --populate archlinux
```

top

Configure System

Configure network

If you want your OS be able to have automatically an IP address for your router or the dhcp server of your network, it is import to activate the dhcpd service at the system startup

```
# systemctl enable dhcpcd
Created symlink /etc/systemd/system/multi-user.target.wants/dhcpcd.service →
/usr/lib/systemd/system/dhcpcd.service.
```

Locale and Language

We can now select the default language of the system by editing the `/etc/locale.gen` file

```
sh-4.3# nano /etc/locale.gen
```

On the list of languages, uncomment the one that you need. I will uncomment `en_US.UTF-8 UTF-8`.

Now we need generate the locale information with the command below

```
sh-4.3# locale-gen
```

Now we must set the **LANG** variable in `/etc/locale.conf`

```
sh-4.3# echo LANG=en_US.UTF-8 > /etc/locale.conf
sh-4.3# export LANG=en_US.UTF-8
```

Timezone

Create a symbolic link with your timezone (to check available timezones, see the files/folders in `/usr/share/zoneinfo/`)

```
# ln -sf /usr/share/zoneinfo/America/New_York /etc/localtime
```

Hardware Clock

```
# hwclock --systohc --utc
```

Hostname

I am going to use `linux` as a hostname:

```
# echo linux > /etc/hostname
```

> Change `linux` to your hostname (Computer Name)

After that, open the file `/etc/hosts` and add the following lines to `/etc/hosts`

```
127.0.0.1        localhost.localdomain         localhost
::1              localhost.localdomain         localhost
127.0.1.1        linux.localdomain             linux
```

> **Remember** to change the `myhostname` to your own)

Nameservers

Check the DNS again (using Google DNS). Open `/etc/resolv.conf` and write:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Create root password

To create root password, type

```
sh-4.3# passwd
```

Locating the EFI partition

The first important thing to do for installing **Grub** on **Arch Linux** is to locate the **EFI** partition. Let's run the following command in order to locate this partition:

```
sh-4.3# fdisk -l
```

We need to check the partition marked as **EFI System**

After that, we need to mount this partition, for example, on `/boot/efi`:

```
sh-4.3# mkdir /boot/efi
sh-4.3# mount /dev/sdc1 /boot/efi
```

Install the bootloader

> If you already have another Linux Distribution installed in your system, you can skip this section and update the Grub from that distribution to recognize Arch Linux.
> For example, you can run `sudo update-grub` from Ubuntu to include Arch Linux in its Grub bootloader

In order to install the bootloader, we need to install the following packages `grub`, `os-prober` and `efibootmgr`. Grub is boot manager and os-prober detects if there are other operating systems installed on the system.

```
sh-4.3# pacman -S grub os-prober efibootmgr
```

Now install the grub on the disk using the command below:

```
sh-4.3# grub-install --target=x86_64-efi --efi-directory=/boot/efi --bootloader-id=GRUB
```

Create a `grub.cfg` file using the command below:

```
sh-4.3# mkinitcpio -p linux
```

Now we can generate the grub

```
sh-4.3# grub-mkconfig -o /boot/grub/grub.cfg
```

Reboot the system

Exit `chroot` environment by pressing Ctrl + D or typing `exit`

Unmount system mount points:

```
# umount -R /mnt
```

Reboot system:

```
# reboot
```

> Remember to remove USB stick on reboot

top

Post Installation

Now you're on your successfull Arch Linux installation.

Login with your user and follow the next steps.

Desktop Environment

Now We're gonna install the Window Manager.

I'll show the steps to install Gnome.

First of all, run the installation command with `pacman`:

```
$ sudo pacman -S gnome gnome-extra
```

When the installation finishes, enable `gdm` to be started with system on boot:

```
$ sudo systemctl enable gdm.service
```

Enable `multilib` repository

To enable multilib repository, uncomment the `[multilib]` section in `/etc/pacman.conf`:

```
# nano /etc/pacman.conf
```

Scroll down and un-comment the 'multilib' repo:

```
#[testing]
#Include = /etc/pacman.d/mirrorlist

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

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

#[community-testing]
#Include = /etc/pacman.d/mirrorlist

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

# If you want to run 32 bit applications on your x86_64 system,
# enable the multilib repositories as required here.

#[multilib-testing]
#Include = /etc/pacman.d/mirrorlist
```

```
[multilib]
Include = /etc/pacman.d/mirrorlist

# An example of a custom package repository. See the pacman manpage for
# tips on creating your own repositories.
#[custom]
#SigLevel = Optional TrustAll
#Server = file:///home/custompkgs
```

```
# pacman -Sy
```

Create new user

It is recommended that you create a normal user account

```
# useradd -m -g users -G wheel,storage,power,lp,network,audio,video,optical -s /bin/bash
sayem
```

Now we must change the password for the user with the `passwd` command

```
# passwd sayem
```

We must edit the `/etc/sudoers` file with the `visudo` command to add the new user :

```
# EDITOR=nano visudo
```

Add the new user `sayem ALL=(ALL) ALL` to `/etc/sudoers`

```
##
## User privilege specification
##
sayem ALL=(ALL) ALL
root ALL=(ALL) ALL
```

Network Manager and services

```
$ sudo pacman -Sy networkmanager-dispatcher-ntpd cronie networkmanager network-manager-
applet acpid cups avahi dbus udisks2
```

```
$ systemctl enable NetworkManager.service
$ systemctl start NetworkManager.service
```

```
$ systemctl enable cronie.service
$ systemctl start cronie.service
```

```
$ systemctl enable ntpd.service
$ systemctl start cronie.service
```

```
$ systemctl enable acpid.service
$ systemctl start acpid.service
```

```
$ systemctl enable avahi-daemon.service
$ systemctl start avahi-daemon.service
```

top