

How to install Arch Linux

A Barebones, No-Nonsense Guide

by Rishikesh Giridhar

July 13, 2025

Contents

Chapter 1

Introduction

You Think You Know Linux?

So, you've used Linux before. Maybe Ubuntu. Maybe Mint. Fedora if you're feeling fancy. You've clicked your way through an installer, watched as some wizard held your hand, and then said to yourself, "Hey, I use Linux now."

But here's the thing: you don't.

Those aren't Linux — those are Linux-flavoured illusions. Training wheels bolted onto a tricycle, wrapped in bubble wrap. If you've never written your own 'fstab', never mounted a root partition manually, never booted to a blinking cursor in a live environment and built your entire operating system line by line — then you've merely touched Linux. You've never wielded it.

Enter Arch Linux.

Why Arch? Because You Want Control.

Other distributions try to guess what you want. Arch asks you.

Where others give you a desktop, Arch gives you a shell. Where others offer security through abstraction, Arch offers power through exposure. It is not "user-friendly" in the traditional sense, because its user is not the traditional type. Its user is someone like you — curious, stubborn, tired of the coddling, and ready to truly understand how Linux works.

You don't "install" Arch. You earn it.

Because Minimalism Isn't a Style — It's a Statement

Arch doesn't come with a GUI, and that's not a bug — it's a feature. It installs only what you explicitly tell it to. No junk. No clipboard managers. No software centres. No 15 background services you never asked for. Just the kernel, the essentials, and you.

Want GNOME? Install it. Want i3? Build it. Want nothing but a terminal forever like a proper chad? Stay there.

Arch is as small or as expansive as you want it to be. You're not fighting the system — you're designing it.

“But My Distro Just Works” — Does It, Though?

Ah yes, the phrase that always precedes a help thread titled **“Why did my update break my desktop?”** You see, when your system is built from the ground up, when you know what every daemon, mount point, and config file is doing, then when something breaks, you don’t panic. You fix it.

Because you know it. Because you built it.

And updates? Arch is a rolling release — not some outdated stable-core nonsense. You get the latest kernel, latest drivers, latest software, on your terms. You’re not stuck six months behind waiting for some release freeze to thaw.

Gaslighting? No — We’re Just Honest

Look, if you think your click-and-install distro is “basically the same” as Arch, that’s fine. You’re entitled to your beliefs, however adorably incorrect they might be.

But let’s be real.

When you say “I use Linux” and someone asks “Oh, which distro?”, and you reply “Arch”, the conversation ends. You win. Because everyone knows: the person who uses Arch doesn’t just use Linux — they understand it. They command it.

You’re not someone running Linux.

You’re someone running Arch.

This Book Is Your Rite of Passage

This guide will walk you through every step: from downloading the ISO to booting into your newly forged Arch install. No fluff. No pre-chewed tutorials. Real commands. Real choices. Real Linux.

If you’re not ready to get your hands dirty, turn back now. There are easier paths — full of GUIs and safety nets. But if you’re ready to see the truth behind the terminal, to graduate from “user” to “builder”, then welcome.

This is where it starts.

Chapter 2

Preinstallation

You Don't Just Install Arch. You Prepare For It.

Welcome to the stage before the stage. This is the part no one talks about. The part the “5-minute YouTube Arch install” videos skip right over. But not you. You, brave reader, are here to do it properly.

This chapter is about preparation. Because installing Arch without proper setup is like parachuting with a trash bag — it's bold, but also very stupid.

Let's get into the meat.

Step 1: Downloading the ISO (Yes, Correctly)

You're going to download the official Arch Linux ISO image. Not from some dodgy Reddit mirror. From the source of all truth:

```
https://archlinux.org/download
```

Once you open this link, you'll see a list of mirrors grouped by country. These are maintained by community and university hosts. Find your country, or the one closest to you geographically. For example:

- If you're in India, select one of the IIT mirrors or TUNA (China) if faster.
- If you're in Europe, stick to local NLU or university mirrors (e.g., Germany → TU Dresden).
- In the US? Use kernel.org, MIT, or OSL mirrors.

Once you choose your mirror, download the file named something like:

```
archlinux-YYYY.MM.DD-x86_64.iso
```

Replace ‘YYYY.MM.DD’ with the release date (e.g., ‘2025.07.01’).

Also download the ‘.sig’ file — this is the cryptographic signature used to verify the ISO is not tampered with.

Why We Verify:

Yes, you really should verify the ISO. Here's why:

- The '.sig' file is signed with a GPG key owned by the Arch Linux release manager. - If someone tampered with your ISO (hello, supply chain attacks), this check would fail.

How to Verify:

First, get the key:

```
gpg --keyserver hkps://keyserver.ubuntu.com --recv-keys 9766E084FB0F43D8
```

↳ '9766E084FB0F43D8' is the long-form fingerprint of the official Arch ISO signing key.

Then verify:

```
gpg --verify archlinux-2025.07.01-x86_64.iso.sig
```

If the output says: “

Good signature from "Pierre Schmitz pierre@archlinux.org"

“ You're golden. If not, throw that ISO out like expired milk.

Step 2: Flashing the ISO to a USB Drive

Do not “open the ISO” and copy its files. That's not how bootable drives work.

Option A: Windows — Use Rufus (The Only Acceptable GUI Method)

1. Download Rufus from: <https://rufus.ie> 2. Plug in your USB drive (at least 2GB). 3. Open Rufus → Select your ISO. 4. **Important:** Choose “DD mode” if prompted. Not “ISO mode.” 5. Start.

Option B: Linux — Use dd (aka Disk Destroyer)

```
sudo dd if=archlinux-2025.07.01-x86_64.iso of=/dev/sdX bs=4M status=progress  
→ oflag=sync
```

- 'if=' = input file (your ISO) - 'of=' = output file (your USB device, like '/dev/sdb' — not a partition like '/dev/sdb1') - 'bs=4M' = block size, optimal for speed - 'status=progress' = shows progress - 'oflag=sync' = ensures all writes are flushed to disk

Double check using 'lsblk' to make sure '/dev/sdX' is really your USB drive.

Step 3: Boot Into the USB Drive (Not Your OS)

Insert the USB into your target machine.

Reboot and spam your BIOS/UEFI boot menu key (often 'F12', 'F10', 'DEL', 'ESC', or 'F2').

From the boot menu, select your USB device. If it says “UEFI: Sandisk”, pick that.

If it boots correctly, you'll see:

“ Arch Linux boot menu Boot Arch Linux (x86_64) ”

Select it and hit enter. You should land in a live shell.

If you land in a graphical installer, you downloaded the wrong distro. Reevaluate your life choices.

Step 4: UEFI or Legacy (Spoiler: It Better Be UEFI)

We don't do BIOS installs in the year of our Lord 2025.

Check:

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

If the folder exists, you're booted in UEFI mode. If not, reboot, enter your firmware settings, and enable:

- “UEFI Mode” or “UEFI Only”
- Disable “Legacy Boot” or “CSM”
- Secure Boot: OFF (for now)

Step 5: Connect to the Internet

A. Ethernet (Wired)? Lucky You. You're Already Online.

To test:

```
ping archlinux.org
```

If you get output with response times, you're in.

B. Wi-Fi (You Like Pain)?

First, start the 'iwctl' shell:

```
iwctl
```

Then:

```
device list          # See your wireless interface, e.g., wlan0
station wlan0 scan    # Scans nearby Wi-Fi
station wlan0 get-networks
station wlan0 connect YourWiFiName
```

You'll be prompted for your Wi-Fi password. If it fails, try again. If it keeps failing, check if:

- Your Wi-Fi is 5GHz-only and your adapter is 2.4GHz.
- You typed the password like a potato.
- You need to unblock the device: 'rfkill unblock wifi'

Step 6: Sync the System Clock (Don't Skip This!)

```
timedatectl set-ntp true
```

This enables automatic time sync via NTP (Network Time Protocol). If your clock is wrong, ‘pacman’ will cry, signatures will fail, and you’ll blame Arch instead of yourself.

Check:

```
timedatectl status
```

Look for: “

System clock synchronized: yes

”

If it says “no,” you’re probably offline or behind a captive portal.

Step 7: Identify Your Drives (Don't Wipe Windows Unless You Mean To)

Use:

```
lsblk
```

You’ll see a tree of devices:

NAME	MAJ\MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	232.9G	0	disk	
sda1	8:1	0	512M	0	part	
sda2	8:2	0	100G	0	part	
nvme0n1	259:0	0	476.9G	0	disk	
nvme0n1p1	259:1	0	512M	0	part	
nvme0n1p2	259:2	0	100.0G	0	part	

- ‘sda’, ‘nvme0n1’, etc. are disks - ‘sda1’, ‘nvme0n1p2’ are partitions

You can also run:

```
fdisk -l
```

Look at the sizes and filesystems. Windows partitions often show as ‘NTFS’. You’ll want to install Arch on a separate disk or partition, or yeet the whole drive (with backups, obviously).

You Are Now Ready

Once you’ve:

- Booted via UEFI
- Verified and flashed the Arch ISO
- Connected to the internet
- Synced your clock

- Identified your disks

...you are officially standing at the gates of Arch.

Next up: slicing up your disk with precision using ‘fdisk’ (and not screwing it up).

Chapter 3

Connecting to the Internet

You Can't Install Arch Offline. (Well, Technically You Can, But You Shouldn't.)

The first thing any good Arch wizard does after booting into the live environment is ensure they are online. Why? Because you're about to pull the entire base system from Arch repositories via 'pacstrap', and unless you've memorised every binary ever made (you haven't), you'll need the internet.

This chapter covers how to connect to the internet, whether you're blessed with Ethernet or condemned to suffer through Wi-Fi.

3.1 Wired Ethernet — The True Master Race

Let's start with the easiest, most reliable method: plugging in a cable.

If your system is connected to a router via Ethernet, congratulations. You're probably already online.

Step 1: Test the connection

```
ping archlinux.org
```

You should see a repeating output like:

```
64 bytes from archlinux.org (95.217.163.246): icmp_seq=1 ttl=51 time=128 ms
```

To stop it, hit 'Ctrl + C'.

If this works, you're online. Proceed to the next chapter. Seriously. You're done here.

If not? The cable may be bad, your router may be dumb, or your DHCP isn't assigning IPs. Let's troubleshoot later.

3.2 Wireless — The Path of Pain

Connecting via Wi-Fi is possible — just not as instant. You'll need to use the ****IWD**** (iNet Wireless Daemon) system included in the Arch ISO.

Step 1: Launch IWD interactive shell

```
iwctl
```

This opens the IWD shell. It should look something like:

```
[iwd]#
```

If you see that, you're inside the network manager. If not, you may not have a supported wireless card.

Step 2: List wireless devices

```
device list
```

Look for something like:

```
Device: wlan0  
Type: wifi
```

Take note of the interface name — usually 'wlan0', 'wlp3s0', or 'wlan1'. We'll use it below.

Step 3: Scan for networks

```
station wlan0 scan
```

(Replace 'wlan0' with your actual device name if different.)

Then, list the results:

```
station wlan0 get-networks
```

This will show a list of all nearby SSIDs.

Step 4: Connect to a network

```
station wlan0 connect YourWiFiSSID
```

↳ You'll be prompted for the password.

If it says "Connected", then congratulations — you're online.

Step 5: Exit the shell

```
exit
```

Now test connectivity:

```
ping archlinux.org
```

If you see replies, it worked. If not? Read the next section.

3.3 Common Wi-Fi Failures and Fixes

1. “Operation Failed” or “No Device” in iwctl?

Run:

```
rfkill list all
```

If any line shows “soft blocked: yes” or “hard blocked: yes”, unblock it:

```
rfkill unblock all
```

Then retry ‘iwctl’.

2. iwctl says connected but ping doesn’t work?

Run:

```
ip a
```

If you don’t see an IP address under your Wi-Fi interface, then DHCP failed. Try:

```
systemctl restart iwd
```

Or restart the whole shell:

```
exit  
iwctl
```

Still nothing? Reboot and try again. Sometimes it’s the firmware (especially on Broadcom or Realtek cards).

3.4 Legacy Option: wifi-menu (Only Available on netctl-based ISOs)

Some very old Arch ISOs ship with ‘wifi-menu’:

```
wifi-menu
```

This launches an ncurses-based selector for wireless networks. It’s deprecated and usually not present in modern ISOs, but if your ISO has it, and ‘iwctl’ doesn’t work, give it a go.

3.5 Using a Static IP (Edge Case or Emergency)

If your router is broken, or you’re in a network without DHCP, you can assign a static IP manually.

Example:

```
ip link set wlan0 up  
ip addr add 192.168.1.100/24 dev wlan0  
ip route add default via 192.168.1.1
```

Then set a DNS resolver:

```
echo "nameserver 1.1.1.1" > /etc/resolv.conf
```

Now try ‘ping archlinux.org’.

3.6 Confirming You’re Online

No matter which method you used, always confirm with:

```
ping archlinux.org
```

If that works, you’re ready to install Arch.

If not, no shame in retrying — but do not proceed until this works. No internet = no pacstrap = no Arch.

Chapter 4

Partitioning the Disk

Look Me in the Eye and Tell Me You're Ready to Wipe a Drive

This is it — the point of no return. From here on out, you're not just reading. You're carving your will into silicon. One wrong device name and you might wipe your Windows partition, your family photos, your tax returns, and any shred of dignity.

Which is exactly why we're going to do it right.

4.1 The Boot Mode Dictates Your Partition Style

Before anything else, make absolutely sure you're booted in ****UEFI mode****. Run:

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

If that directory exists — you're good. Proceed with GPT partitioning (the modern standard).

If it doesn't exist, you're in Legacy BIOS mode. That's fine too (just worse). We'll handle that in a separate section.

Wait — What's GPT vs MBR?

- ****GPT (GUID Partition Table)****: Modern. Required for UEFI. Supports more than 4 partitions. Safer and better aligned. - ****MBR (Master Boot Record)****: Ancient BIOS format. Limited. Clunky. Avoid unless your firmware forces it.

Unless your PC is from the Jurassic era, use GPT.

4.2 Identify the Target Disk (Don't Wipe the Wrong One)

Run:

```
lsblk
```

Example output:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPPOINT
sda	8:0	0	232.9G	0	disk	
nvme0n1	259:0	0	476.9G	0	disk	
nvme0n1p1	259:1	0	512M	0	part	/boot
nvme0n1p2	259:2	0	476G	0	part	/

Look at the SIZE column. If you're installing onto a blank SSD, it's probably `/dev/sda` or `/dev/nvme0n1`.

If you're unsure — stop. Double check. Remove all drives except the one you want to install on.

4.3 Partition Layout (UEFI Scheme)

For a typical UEFI install, here's what we want:

- `/dev/sdX1` → EFI System Partition (512MB, FAT32)
- `/dev/sdX2` → Root (`/`) partition (rest of the disk, ext4)

Optional `/dev/sdX3` → Swap or Home

No need to overcomplicate it. One root partition is all Arch needs.

4.4 Using fdisk (Beginner-Friendly, Deadly If You Panic)

Start `fdisk`:

```
fdisk /dev/sdX
```

Replace `/dev/sdX` with your actual disk. NOT a partition (e.g. NOT `/dev/sda1`).

Step-by-step fdisk (GPT Mode):

1. Press `g` → creates a new GPT partition table (wipes the drive) 2. Press `n` → create new partition - Partition number: 1 - First sector: press Enter - Last sector: `+512M` - Type: `t`, then type code: `1` (EFI System) 3. Press `n` again → create root partition - Partition number: 2 - Accept defaults for rest of disk 4. (Optional) Press `n` again → swap partition - Set size like `+8G` - Set type to `19` (Linux swap)

Once done:

```
w
```

`w` = write changes and exit. There is no undo.

4.5 Using cgdisk (For Visual Learners Who Like Retro Interfaces)

```
cgdisk /dev/sdX
```

You'll get a TUI interface. It's self-explanatory:

- Use arrows to select “New” - Type partition sizes like ‘512M’, ‘100- Set type codes: - ‘EF00’ for EFI System - ‘8300’ for Linux filesystem - ‘8200’ for Linux swap

Write changes and exit.

4.6 Formatting the Partitions

Now that you’ve sliced the disk, time to give them filesystems.

EFI Partition (FAT32):

```
mkfs.fat -F32 /dev/sdX1
```

Root Partition (ext4):

```
mkfs.ext4 /dev/sdX2
```

Optional: Swap Partition

Create and activate it:

```
mkswap /dev/sdX3
swapon /dev/sdX3
```

Note: Swap is optional. With 8GB+ RAM, you don’t really need it unless you plan to hibernate.

4.7 Mounting the Filesystems

Arch doesn’t do this for you — you must mount everything manually.

```
mount /dev/sdX2 /mnt
mkdir /mnt/boot
mount /dev/sdX1 /mnt/boot
```

If you created a separate home partition:

```
mkdir /mnt/home
mount /dev/sdX4 /mnt/home
```

At this point, your disk layout should look like this:

```
/
    boot          mounted from /dev/sdX1
    (root)        mounted from /dev/sdX2
    home          optionally /dev/sdX4
    swap          activated via swapon
```

4.8 Verify Mounts

Run:

```
lsblk -f
\mount | grep mnt
```

Make sure all mount points are correct. If not, unmount and try again.

4.9 If You're Using BIOS (MBR) Boot Instead...

Rare, but here's the rundown:

- Use 'fdisk' with 'o' to create a DOS partition table - First partition: root ('/') → type 83 (Linux) - Optional: second partition for swap → type 82 - No EFI partition needed - Format and mount as above

Note: You will use GRUB as your bootloader instead of systemd-boot.

4.10 Final Checkpoint

Drive is partitioned Filesystems are created Mount points are ready Swap is active (if used)
You didn't nuke the wrong disk

You're ready to 'pacstrap' this beast into existence.

Chapter 5

Installing the Base System

Time to Put Flesh on the Skeleton

You’ve prepped the disk. You’ve mounted your partitions. `/mnt` is waiting — hollow, empty, full of possibility.

Now comes the moment Arch veterans call “pacstrapping.” This is where you install the base packages that form the foundation of your Linux system.

You’re not clicking checkboxes. You’re calling down the barest, rawest, most essential components of an operating system — one that does exactly what you tell it to and absolutely nothing else.

5.1 What is pacstrap?

`pacstrap` is a special Arch install tool. It installs packages into a target root directory — in our case, `/mnt`.

So, when you run:

```
pacstrap -K /mnt base linux linux-firmware
```

Here’s what each part does:

- `-K`: Copies your current machine’s package signing keys to the target system (saves you a step later).
- `/mnt`: Target directory for the installation — you previously mounted your root partition here.
- `base`: A metapackage that pulls in core utilities (`bash`, `coreutils`, `pacman`, `systemd`, etc.).
- `linux`: Installs the latest Arch kernel (you can swap this for `linux-lts` or `linux-zen` later).
- `linux-firmware`: Adds firmware blobs for hardware like Wi-Fi, audio, GPUs, etc.

5.2 Minimal Example — Barebones Install

```
pacstrap -K /mnt base linux linux-firmware
```

This is enough to boot, configure, and later install your user environment.

5.3 Commonly Added Packages (Optional but Useful)

You can also throw in:

```
pacstrap -K /mnt base linux linux-firmware vim nano man-db man-pages texinfo
```

Explanation:

- ‘vim’ or ‘nano’: Text editors. Choose your pain. - ‘man-db’, ‘man-pages’, ‘texinfo’: Documentation. So you don’t have to Google how ‘mount’ works at 3AM.

Want a network manager? You’ll install that later, **inside** the chroot. Don’t worry about it now.

5.4 What /mnt Is, Really

Right now, ‘/mnt’ is where your future system lives. Every command you give from this point forward that mentions ‘/mnt’ is configuring your soon-to-be root filesystem.

When you chroot into it, ‘/mnt’ will become ‘/’. You’ll be inside your new system. But for now, we’re still outside — gently crafting its skeleton.

5.5 Creating the fstab — The System’s Map of Your Disk

‘fstab’ tells the system how to mount disks and partitions automatically at boot.

To generate it:

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

- ‘-U’: Uses UUIDs (universally unique identifiers) instead of ‘/dev/sdX’ — avoids confusion when device names change. - ‘>>’: Appends output to the fstab file. - ‘/mnt/etc/fstab’: The config file that gets created.

Verify It:

Just to be safe, always check the file:

```
cat /mnt/etc/fstab
```

You should see entries for ‘/’, ‘/boot’, and ‘swap’ (if used), like:

```
/dev/nvme0n1p2 UUID=abc123... / ext4 defaults 0 1
/dev/nvme0n1p1 UUID=def456... /boot vfat defaults 0 2
```

If anything looks wrong — redo the mount and generate fstab again.

5.6 Enter the Void: arch-chroot

Now it’s time to step into your new system and take control.

```
arch-chroot /mnt
```

You'll land in what looks like your usual shell, but you're now **inside** your installed system. You can configure it just like any regular Arch install from here.

From this point forward, any changes you make (like setting passwords, locale, hostname, installing packages) will be to your new system — not the live ISO.

Welcome home, user. You've built your skeleton.

Checklist Before Moving On

'pacstrap' ran without errors 'fstab' was generated and looks correct You can 'arch-chroot' into the new system You understand '/'mnt' is your root-to-be

If all that's done, you're ready to give this beast a soul — hostname, locale, time, users, boot-loader, the lot. But that's in the next chapter.

Chapter 6

Configuring the System

6.1 Fstab

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

6.2 Chroot and Time

```
arch-chroot /mnt  
ln -sf /usr/share/zoneinfo/Asia/Kolkata /etc/localtime  
hwclock --systohc
```

6.3 Locale

```
vim /etc/locale.gen  
# Uncomment: en_US.UTF-8 UTF-8  
locale-gen  
echo "LANG=en_US.UTF-8" > /etc/locale.conf
```

6.4 Hostname

```
echo "archmachine" > /etc/hostname
```


Chapter 7

Bootloader

Install ‘systemd-boot’:

```
bootctl install
```

/boot/loader/entries/arch.conf:

```
title    Arch Linux
linux    /vmlinuz-linux
initrd   /initramfs-linux.img
options  root=PARTUUID=...
```


Chapter 8

Final Steps

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


Chapter 9

After Reboot

You now have a minimal Arch Linux installation! Configure your user, networking, packages, and (if desired) a desktop environment next.