HOW TO INSTALL

archlinux™

BY RISHIKESH GIRIDHAR

How to install Arch Linux

A Barebones, Cheesy, yet No-Nonsense Guide

by Rishikesh Giridhar

Contents

1 Introduction **3**
You Think You Know Linux? . . . 3
Why Arch? Because You Want Control. . . . 3
Because Minimalism Isn't a Style — It's a Statement . . . 3
"But My Distro Just Works" — Does It, Though? . . . 4
Gaslighting? No — We're Just Honest . . . 4
This Book Is Your Rite of Passage . . . 4

2 Preinstallation **5**
You Don't Just Install Arch. You Prepare For It. . . . 5
Step 1: Downloading the ISO (Yes, Correctly) . . . 5
Step 2: Flashing the ISO to a USB Drive . . . 6
Step 3: Boot Into the USB Drive (Not Your OS) . . . 6
Step 4: UEFI or Legacy (Spoiler: It Better Be UEFI) . . . 7
Step 5: Connect to the Internet . . . 7
Step 6: Sync the System Clock (Don't Skip This!) . . . 8
Step 7: Identify Your Drives (Don't Wipe Windows Unless You Mean To) . . . 8
You Are Now Ready . . . 8

3 Connecting to the Internet **10**
You Can't Install Arch Offline. (Well, Technically You Can, But You Shouldn't.) . . . 10
3.1 Wired Ethernet — The True Master Race . . . 10
3.2 Wireless — The Path of Pain . . . 10
3.3 Common Wi-Fi Failures and Fixes . . . 12
3.4 Legacy Option: wifi-menu (Only Available on netctl-based ISOs) . . . 12
3.5 Using a Static IP (Edge Case or Emergency) . . . 12
3.6 Confirming You're Online . . . 13

4 Partitioning the Disk **14**
Look Me in the Eye and Tell Me You're Ready to Wipe a Drive . . . 14
4.1 The Boot Mode Dictates Your Partition Style . . . 14
4.2 Identify the Target Disk (Don't Wipe the Wrong One) . . . 14
4.3 Partition Layout (UEFI Scheme) . . . 15
4.4 Using `fdisk` (Beginner-Friendly, Deadly If You Panic) . . . 15
4.5 Using `cgdisk` (For Visual Learners Who Like Retro Interfaces) . . . 15
4.6 Formatting the Partitions . . . 16
4.7 Mounting the Filesystems . . . 16
4.8 Verify Mounts . . . 16
4.9 If You're Using BIOS (MBR) Boot Instead... . . . 17
4.10 Final Checkpoint . . . 17

5 Installing the Base System **18**
5.1 What is `pacstrap`? . . . 18
5.2 Minimal Example — Barebones Install . . . 18
5.3 Commonly Added Packages (Optional but Useful) . . . 19
5.4 What `/mnt` Is, Really . . . 19
5.5 Creating the `fstab` — The System's Map of Your Disk . . . 19
5.6 Enter the Void: `arch-chroot` . . . 19

6 System Configuration **21**
6.1 Set the Time Zone . . . 21
6.2 Set Locale (Language + Encoding) . . . 21
6.3 Set the Hostname . . . 22
6.4 Configure /etc/hosts (Networking ID) . . . 22
6.5 Set Root Password . . . 22
6.6 Create a Non-Root User . . . 22
6.7 Enable `sudo` for Wheel Group . . . 23
6.8 Install Networking Tools . . . 23
6.9 Install Text Editor (Because You'll Need One) . . . 23
6.10 Generate initramfs (Boot Image) . . . 24

7 Bootloader Setup **25**
7.1 Option A: systemd-boot (UEFI Only, Arch Approved) . . . 25
7.2 Option B: GRUB (UEFI or Legacy BIOS) . . . 26
7.3 Checkpoint: Ready for Reboot? . . . 27

8 Post-Install Essentials **28**
8.1 Update Everything . . . 28
8.2 Install CPU Microcode (Do Not Skip) . . . 28
8.3 Audio — Because Silence is for Windows Users . . . 29
8.4 Graphics Drivers — Know Your GPU . . . 29
8.5 Bluetooth (If You Need It) . . . 29
8.6 Printing (For Those Who Still Use Paper) . . . 30
8.7 Network Tools (Optional, But Helpful) . . . 30
8.8 Basic Terminal + GUI Tools . . . 30
8.9 Optional: Enable Reflector for Faster Mirrors . . . 30
8.10 Checkpoint Before Graphical Setup . . . 30

9 Graphical Environment Setup **31**
9.1 Wayland is Now . . . 31
9.2 Installing Wayland (if not bundled) . . . 31
9.3 Installing a Desktop Environment (Wayland-First Choices) . . . 31
9.4 Display Manager vs startx . . . 32
9.5 Fonts and Themes . . . 32
9.6 Post-DE Checklist . . . 33

10 Precepts of the Arch Faithful **34**
10.1 1. The Kernel — The Soul of the System . . . 34
10.2 2. initramfs — The Bridge to the Kernel . . . 34
10.3 3. systemd — The Grand Orchestrator . . . 35
10.4 4. Shell and Terminal — Your First Interface . . . 35
10.5 5. Display Server — How Graphics Get Drawn . . . 35

10.6 6. Desktop Environment — The Furniture of Your OS 36
10.7 7. How Arch Works — Rolling, Minimal, Transparent 36
10.8 8. Why the Install is Manual (And Why It Matters) 36

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 'iwd' 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 MOUNTPOINT
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, bootloader, the lot. But that's in the next chapter.

Chapter 6

System Configuration

You've Built the Flesh. Now It Needs a Brain.

You've chrooted into your fresh new Arch system. It knows nothing. It has no sense of time, no language, no name, no user. If you were to reboot now, you'd have a blank disk with a kernel that doesn't even know it exists.

Let's fix that.

6.1 Set the Time Zone

Start by setting the system's time zone. You don't want your logs timestamped in UTC+Mars.

Find your time zone:

```
ls /usr/share/zoneinfo
```

This directory contains all time zones. Drill down into your region and city, e.g.:

```
ls /usr/share/zoneinfo/Asia
```

Once you know yours (e.g., 'Asia/Kolkata'), set it:

```
ln -sf /usr/share/zoneinfo/Asia/Kolkata /etc/localtime
```

Sync the hardware clock:

```
hwclock --systohc
```

This sets the hardware clock (BIOS clock) to match the system time.

6.2 Set Locale (Language + Encoding)

Edit '/etc/locale.gen' to uncomment the locales you want.

```
nano /etc/locale.gen
```

Find the line:

```
#en_US.UTF-8 UTF-8
```

And remove the ‘‘:

```
en_US.UTF-8 UTF-8
```

Save and exit (‘Ctrl+O‘, ‘Enter‘, ‘Ctrl+X‘).

Then generate the locales:

```
locale-gen
```

Set default system language:

Create or edit ‘/etc/locale.conf‘:

```
echo "LANG=en_US.UTF-8" > /etc/locale.conf
```

6.3 Set the Hostname

Pick a name for your machine. This will appear in your shell prompt, your SSH sessions, and every log file. It’s your system’s name in the realm.

```
echo "archbeast" > /etc/hostname
```

6.4 Configure /etc/hosts (Networking ID)

Edit your hosts file to match your hostname:

```
nano /etc/hosts
```

Add these lines:

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

Replace ‘archbeast‘ with your actual hostname.

6.5 Set Root Password

```
passwd
```

Enter and confirm your root password. Choose wisely. You’re root — the god of this system.

6.6 Create a Non-Root User

Root is powerful — too powerful. You’ll want a regular user.

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

This creates a user named ‘rishi‘: - ‘-m‘: makes a home directory - ‘-G wheel‘: adds to the ‘wheel‘ group (for sudo access) - ‘-s /bin/bash‘: sets the shell

Set the user’s password:

```
passwd rishi
```

6.7 Enable sudo for Wheel Group

First, install sudo:

```
pacman -S sudo
```

Then edit the sudoers file with:

```
EDITOR=nano visudo
```

Find the line:

```
# %wheel ALL=(ALL:ALL) ALL
```

Uncomment it (remove the ‘‘).

Now any user in the ‘wheel‘ group can use ‘sudo‘.

6.8 Install Networking Tools

Let’s install tools so you can connect to the internet after reboot.

```
pacman -S networkmanager
```

Enable it for boot:

```
systemctl enable NetworkManager
```

This gives you Wi-Fi/Ethernet management out of the box.

6.9 Install Text Editor (Because You’ll Need One)

Pick your poison:

```
pacman -S vim
```

or

```
pacman -S nano
```

Nano is easier, Vim is better (if you know it).

6.10 Generate initramfs (Boot Image)

This step generates the initramfs — a compressed archive that boots your kernel and mounts the root file system.

Usually this happens automatically with ‘pacstrap‘, but let’s be sure:

```
mkinitcpio -P
```

That’s it. Your system is now configured to boot and function.

Next up? The bootloader. Yes, the thing that makes your system show up when you turn it on.

Chapter 7

Bootloader Setup

This Is Where the Magic Happens

You've got a fully configured root filesystem, a working kernel, and an initramfs image ready to roll. But if you reboot now? Nothing happens.

That's because we haven't installed the bootloader — the sacred whisperer that tells your firmware where to find your OS.

Let's fix that.

7.1 Option A: systemd-boot (UEFI Only, Arch Approved)

This is the cleanest, fastest bootloader for UEFI systems. It integrates tightly with systemd and doesn't carry GRUB's bloaty legacy.

Step 1: Ensure You're in UEFI Mode

Just double check, again:

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

If it's empty or missing, stop — you need to use GRUB instead (jump to Section B).

Step 2: Install systemd-boot

```
bootctl install
```

This installs ‘systemd-boot‘ into your EFI system partition (‘/boot‘), assuming you mounted ‘/dev/sdX1‘ to ‘/boot‘ earlier (as instructed).

You should see:

```
Created "/boot/EFI/systemd".
Copied "/usr/lib/systemd/boot/efi/systemd-bootx64.efi"     "/boot/EFI/systemd
    ↪ /systemd-bootx64.efi".
Created EFI boot entry.
```

Step 3: Create the Loader Configuration File

Create ‘/boot/loader/loader.conf‘:

```
nano /boot/loader/loader.conf
```

Add the following:

```
default arch
timeout 3
console-mode max
editor no
```

- ‘default arch‘: Sets which entry boots by default - ‘timeout‘: How long the bootloader waits before auto-boot - ‘console-mode max‘: Makes the font less potato - ‘editor no‘: Disables editing the boot entry mid-boot

Step 4: Create the Boot Entry

Create a file at ‘/boot/loader/entries/arch.conf‘:

```
nano /boot/loader/entries/arch.conf
```

Paste this:

```
title   Arch Linux
linux   /vmlinuz-linux
initrd  /initramfs-linux.img
options root=UUID=your-root-uuid-here rw
```

Now get your root partition’s UUID:

```
blkid
```

Find the line for your root partition (e.g., ‘/dev/sda2‘) and copy its UUID. Replace ‘your-root-uuid-here‘ with it (keep the UUID in the same format).

Your final entry might look like:

```
options root=UUID=abc1234-5678-dead-beef-1234567890ab rw
```

You’re done. That’s it. When you reboot, systemd-boot will pick up your entry and boot Arch.

7.2 Option B: GRUB (UEFI or Legacy BIOS)

If you’re stuck with BIOS, or you dual-boot Windows, or your firmware is allergic to simplicity, you’ll need GRUB.

Step 1: Install GRUB and Required Tools

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

- ‘efibootmgr‘: Required for UEFI - ‘os-prober‘: Detects other OSes (Windows, other Linux distros) - ‘mtools‘, ‘dosfstools‘: Required to work with FAT partitions (EFI)

Step 2A: UEFI GRUB Installation

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

Step 2B: BIOS GRUB Installation

```
grub-install --target=i386-pc /dev/sdX
```

Where ‘/dev/sdX‘ is your disk, *not* a partition (e.g., ‘/dev/sda‘ not ‘/dev/sda1‘).

Step 3: Generate GRUB Config

```
grub-mkconfig -o /boot/grub/grub.cfg
```

You’ll see it scanning for kernels and other OSes. If you installed ‘os-prober‘, it might pick up Windows or another Linux install.

And with that, you have a bootloader.

7.3 Checkpoint: Ready for Reboot?

Make sure:

- systemd-boot or GRUB is installed - You’ve created at least one boot entry - ‘blkid‘ shows UUIDs and they match your config - Your boot partition is correctly mounted

Now… exit and unmount:

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

Then finally:

```
reboot
```

If all goes well, your screen will flash black, your heart will skip, and then you’ll see the **Arch Linux** bootloader screen.

Welcome to the club.

Chapter 8

Post-Install Essentials

It Boots. Now Let's Make It Functional.

Congrats. If you've made it this far, you've done more than 90% of the world ever will. Your Arch system boots. It breathes. But it doesn't yet *live*.

This chapter is about giving it the basic tools it needs to become a daily driver. We're not talking KDE Plasma or i3 just yet — we're talking **drivers**, **microcode**, **audio**, and a few **sane defaults**.

8.1 Update Everything

Even though 'pacstrap' installed the latest snapshot, package updates can happen hourly. Best to start clean.

```
pacman -Syu
```

This updates all packages to the current latest from the Arch mirrors.

8.2 Install CPU Microcode (Do Not Skip)

Microcode is firmware for your CPU. Yes, even the brain of your machine ships buggy code. We fix that here.

For Intel CPUs:

```
pacman -S intel-ucode
```

Then edit your 'arch.conf' (if using systemd-boot):

```
nano /boot/loader/entries/arch.conf
```

Add this **above** the 'initrd /initramfs-linux.img' line:

```
initrd /intel-ucode.img
```

For AMD CPUs:

```
pacman -S amd-ucode
```

Same edit applies in your boot entry.

This ensures your CPU gets updated microcode at boot — crucial for stability and performance.

8.3 Audio — Because Silence is for Windows Users

Arch uses ‘pipewire‘ — the superior modern replacement for PulseAudio.

```
pacman -S pipewire pipewire-alsa pipewire-pulse pipewire-jack wireplumber
```

Then enable the audio service:

```
systemctl enable --now pipewire pipewire-pulse wireplumber
```

Test later with:

```
speaker-test -c 2
```

8.4 Graphics Drivers — Know Your GPU

Without a graphics driver, your system will feel like a potato in a freezer.

Intel (Integrated Graphics):

```
pacman -S xf86-video-intel mesa vulkan-intel libva-intel-driver
```

AMD (Radeon/VEGA):

```
pacman -S xf86-video-amdgpu mesa vulkan-radeon libva-mesa-driver
```

NVIDIA (Dedicated Cards):

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

Also run:

```
nvidia-modprobe
```

¿ You don’t need ‘xf86-video-vesa‘. That’s a fallback and usually worse.

8.5 Bluetooth (If You Need It)

```
pacman -S bluez bluez-utils
systemctl enable --now bluetooth
```

Use ‘bluetoothctl‘ for pairing devices from terminal.

8.6 Printing (For Those Who Still Use Paper)

```
pacman -S cups system-config-printer
systemctl enable --now cups
```

CUPS is the printing daemon. Works with most USB/WiFi printers via GUI once DE is installed.

8.7 Network Tools (Optional, But Helpful)

```
pacman -S wget curl git
```

Useful for fetching things, cloning dotfiles, and surviving in general.

8.8 Basic Terminal + GUI Tools

File Manager (CLI):

```
pacman -S ranger
```

Terminal Emulators:

```
pacman -S alacritty kitty
```

Fonts:

```
pacman -S ttf-dejavu ttf-liberation noto-fonts
```

Trust me — install fonts before your DE. Otherwise half the UI will look like corrupted boxes.

8.9 Optional: Enable Reflector for Faster Mirrors

Faster mirrors = faster installs/updates. Install and enable reflector:

```
pacman -S reflector
```

Then enable the mirror ranking service:

```
systemctl enable reflector.timer
```

You can also manually update mirrors:

```
reflector --latest 10 --protocol https --sort rate --save /etc/pacman.d/
    ↪ mirrorlist
```

8.10 Checkpoint Before Graphical Setup

System updated Microcode installed Audio working via pipewire Graphics drivers installed Bluetooth and printing services ready Fonts and terminal tools in place

You're now ready for a display server and a desktop environment — a throne for your new OS.

Chapter 9

Graphical Environment Setup

CLI is Pure, But GUI is Useful

Yes, the terminal is a sacred rite. But there comes a time to bless your Arch system with **vision** — a graphical environment.

In this chapter:

1. We'll install a display server (Wayland-first, X11 optional)
2. Choose a desktop environment or window manager
3. Set up a display/login manager or 'startx'

9.1 Wayland is Now

Wayland is no longer the future — it's the present. It's the default for GNOME, KDE Plasma, and others.

X11 is still supported, but it's only used:

- For legacy apps or WMs
- When Wayland has known compatibility issues

Unless you have a specific need, stick to **Wayland**.

9.2 Installing Wayland (if not bundled)

Most DEs (GNOME, KDE) bring Wayland along. But to be explicit:

```
pacman -S wayland xdg-desktop-portal xdg-desktop-portal-wlr
```

Note: Some portals are DE-specific — GNOME/KDE pull their own.

9.3 Installing a Desktop Environment (Wayland-First Choices)

Option A: KDE Plasma 6 (Now Wayland by Default)

```
pacman -S plasma-meta kde-applications qt6-wayland
pacman -S sddm
systemctl enable sddm
```

Wayland will be default. If you ever want to switch to X11, just select it from the SDDM session menu.

Option B: GNOME (Wayland Default)

```
pacman -S gnome gnome-extra
systemctl enable gdm
```

GDM launches into Wayland by default. No config needed.

Option C: Sway (Wayland-native i3 Alternative)

```
pacman -S sway alacritty foot wl-clipboard xwayland
```

Sway is a tiling window manager for Wayland. ‘xwayland‘ allows legacy X apps to run.

You can launch Sway from TTY:

```
exec sway
```

Option D: XFCE (X11 Only, for Now)

```
pacman -S xfce4 xfce4-goodies
pacman -S lightdm lightdm-gtk-greeter
systemctl enable lightdm
```

Still stuck on X11, but extremely lightweight.

9.4 Display Manager vs startx

- A **Display Manager** gives you a GUI login screen. - ‘startx‘ (X11) or ‘exec sway‘ (Wayland) are CLI launch options.

If you’re using GNOME or KDE: use GDM or SDDM. If using a Wayland WM like Sway, you can launch manually from TTY.

Make GUI login default:

```
systemctl set-default graphical.target
```

9.5 Fonts and Themes

Always install fonts before launching a DE:

```
pacman -S ttf-dejavu ttf-liberation noto-fonts ttf-nerd-fonts-symbols
```

GTK/Qt Theme config:

```
pacman -S lxappearance qt5ct qt6ct
```

9.6 Post-DE Checklist

Wayland installed and used by default GNOME/KDE auto-launch with GDM/SDDM Fonts and themes configured Display target set to graphical You can launch into GUI on boot (or via TTY for Sway)

You now wield a **Wayland-powered** desktop Arch system. You didn't just install an OS — you cultivated a kingdom.

Chapter 10

Precepts of the Arch Faithful

Before You Install, Understand What You're Installing

Most people blindly click "Next" through installers and never know what makes their system tick. But Arch Linux doesn't do *"Next"*.

This chapter is your **deep dive** into what makes your system **boot, breathe, draw, and obey**.

10.1 1. The Kernel — The Soul of the System

At its heart, Linux is **just the kernel** — the core software that talks to your hardware and manages your system's resources.

The kernel does things like:

- Handling memory (RAM allocation)
- Scheduling CPU tasks
- Managing I/O (input/output to devices)
- Enabling hardware features (Wi-Fi, USB, Graphics, etc.)

In Arch, the default kernel is 'linux', but you can also choose others:

- 'linux-lts' – Long-term support
- 'linux-zen' – Optimised for performance
- 'linux-hardened' – Enhanced security

The kernel is stored in '/boot/vmlinuz-linux' and gets loaded at boot time via your bootloader (systemd-boot, GRUB, etc.)

10.2 2. initramfs — The Bridge to the Kernel

When your system boots, it doesn't jump straight into the root file system. It first loads a temporary one: **initramfs** (Initial RAM Filesystem).

This small image:

- Loads essential drivers (like filesystem, encryption, LVM)
- Mounts the real root partition
- Hands control over to the actual system (‘/‘)

It’s created using ‘mkinitcpio‘, and lives at ‘/boot/initramfs-linux.img‘.

10.3 3. systemd — The Grand Orchestrator

Once your kernel and root filesystem are live, you need someone to **start services**, manage your login sessions, handle daemons, and coordinate boot stages.

That’s where ‘systemd‘ comes in.

It replaces the old SysV-style init system and:

- Manages services like networking, display manager, audio, etc.
- Handles dependencies (e.g., NetworkManager starts before KDE)
- Provides logging via ‘journalctl‘
- Supports parallel boot for faster startup

You’ll use it all the time:

```
systemctl enable bluetooth
systemctl start NetworkManager
systemctl set-default graphical.target
```

10.4 4. Shell and Terminal — Your First Interface

The shell is the interface that interprets your commands. It’s not the terminal itself — it’s the program running *inside* the terminal.

- ‘bash‘ – Default shell, most common
- ‘zsh‘ – Fancy shell with themes and plugins
- ‘fish‘ – User-friendly shell with autosuggestions

The **terminal emulator** is the GUI program that gives you access to the shell — like ‘alacritty‘, ‘gnome-terminal‘, ‘konsole‘, etc.

10.5 5. Display Server — How Graphics Get Drawn

You can’t show windows without a display server.

X11 (xorg)

The original. Decades old. Works fine. Supported everywhere. But:

- No security between apps
- Janky multi-monitor support
- Doesn’t scale well on HiDPI screens

Wayland (Modern Default)

The new standard. Used by GNOME, KDE, Sway, Hyprland.

- Per-app security and isolation
- Tear-free rendering
- Fractional scaling works properly
- Compositing is built-in

Arch supports both — but Wayland is the **default for modern systems**.

10.6 6. Desktop Environment — The Furniture of Your OS

A Desktop Environment (DE) is a full suite of software that gives you a graphical experience:

- A window manager (to place and move windows)
- Panels, app launchers, system tray, settings
- Themes, compositors, wallpapers, login screen

Examples:

- **GNOME** — Opinionated, Wayland-first, great out of the box
- **KDE Plasma** — Customisable to death, fast and beautiful
- **XFCE** — Lightweight and no-nonsense
- **i3/Sway** — Tiling WMs for keyboard gods

You can even build your own DE from just a WM + panel + launcher.

10.7 7. How Arch Works — Rolling, Minimal, Transparent

Arch Linux has no versions. You install once, update forever.

- Rolling release model: updates are constant, not periodic
- No default DE: you install only what you want
- Pacman: powerful package manager
- AUR: community-contributed packages (via ‘yay‘, ‘paru‘, etc.)
- ArchWiki: best documentation on Earth (literally)

There are no defaults, no assumptions. You are the architect.

10.8 8. Why the Install is Manual (And Why It Matters)

You’ve already noticed: installing Arch is not plug-and-play.

That's by design.

- You learn every step — from partitioning to bootloading
- You know exactly what's installed (no bloat)
- You fix your own system because you understand it

Once installed, your Arch system is:

- Minimal — nothing you didn't ask for
- Bleeding edge — always the latest Linux tech
- Totally yours — configured how *you* want