

Von der Konsole zum performanten Xfce-Desktop: Arch Linux

 Anleitung als PDF anzeigen oder herunterladen

Der Arch-Profi-Guide: Eine ausführliche Anleitung für Einsteiger und Fortgeschrittene

Stand: 18. März 2026

Arch Linux Installation 2026: Modern, puristisch und pfeilschnell























Erlebe die volle Kontrolle über dein System. Diese Anleitung führt dich Schritt für Schritt von der Konsole zum performanten Xfce-Desktop. Optimiert für moderne AMD-Hardware (AM5/RX 7000), nutzt dieser Guide das Btrfs-Dateisystem und den effizienten EFISTUB-Bootloader für maximale Performance ohne unnötigen Ballast.

Das erwartet dich in dieser Anleitung:

Diese Anleitung ist speziell für **moderne AMD AM5-Systeme** (Ryzen 7000/8000/9000) und aktuelle **AMD Radeon-Grafikkarten** optimiert. Wir bauen ein System, das nicht nur schnell ist, sondern auch technisch auf dem neuesten Stand:

- **Puristischer Boot:** Kein GRUB, kein Ballast. Wir nutzen **EFISTUB**, um den Kernel direkt vom MSI-Mainboard aus zu starten – für rekordverdächtige Bootzeiten.
- **Modernes Dateisystem:** Volle **Btrfs-Unterstützung** inklusive Optimierungen für NVMe-SSDs und Datensicherheit.
- **OEM-Look & Feel:** Ein flackerfreier Systemstart mit deinem **MSI-Logo** und dem Arch-Ladekreis dank Plymouth und Early KMS.
- **Maximale Stabilität:** Spezifische BIOS-Fixes für AM5 (Idle-Stability) und CPU-Optimierung via **Curve Optimizer**.
- **Sicherheits-Garantie:** Erstellung eines **fix-boot.sh** Rettungs-Skripts, mit dem du deinen Bootloader nach jedem BIOS-Update in Sekunden reparieren kannst.

Übersicht

- .. **a.** Einleitung
 - .. **b.** Setup-Checkliste: Voraussetzungen für das Arch-System
 - .. **c.** BIOS/UEFI Vorbereitung für aktuelle MSI AM5 Motherboards
 - .. **d.** Das Installations-Medium: Arch Linux startklar machen
 - .. **e.** Willkommen in der Konsole: Die Arbeitsumgebung vorbereiten
 - .. **f.** Verbindung steht? Netzwerk- und Zeitabgleich
 - .. **g.** Das Fundament: Partitionierung einer NVMe SSD
 - .. **h.** Dateisysteme: Btrfs-Optimierung und Mount-Struktur
 - .. **i.** Server-Tuning: Schnelle Spiegelserver für den Download
 - .. **j.** Das Rohbausystem | Installation der Basis-Pakete mit Pacstrap
 - .. **k.** fstab: Laufwerke dauerhaft verankern
 - .. **l.** arch-chroot | Der Wechsel in das neue System
 - .. **m.** Identität und Sicherheit: Hostname, Benutzer und Root-Rechte
 - .. **n.** System-Sprache und Regionaleinstellungen
 - .. **o.** Autopilot aktivieren: Paket-Tuning und Mirror-Automatisierung
 - .. **p.** Desktop-Vorbereitung: System-Dienste | Sound (PipeWire) | Hardware
 - .. **q.** Grafik-Power: X-Server, AMD-Treiber und Early KMS
 - .. **r.** Desktop-Komfort: LightDM, Xfce und der Autologin
 - .. **s.** Dienste und Hintergrundprozesse aktivieren
 - .. **t.** Der Zündschlüssel: UEFI-Direktboot (EFISTUB)
 - .. **u.** Optional: Dein MSI- & Arch-Logo im Rampenlicht
 - .. **v.** Der Neustart – Dein System geht live
-

a. Einleitung

Die Installation von Arch Linux ist kein Hexenwerk. Zwar bietet die Distribution mittlerweile einen automatisierten Installer, doch dieser Guide verfolgt einen anderen Weg: Wir behalten die volle Kontrolle über Dienste, Treiber und die Partitionierung, indem wir die gesamte Installation manuell in der Konsole durchführen.

Viele Anleitungen im Netz sind leider nicht „aus einem Guss“. Dieser Anfänger-Guide ist das Ergebnis jahrelanger Pflege und kontinuierlicher Überarbeitung. Basierend auf der Videoreihe von *unicks.eu* und dem offiziellen Arch-Wiki, ist diese Dokumentation unter Einbeziehung moderner KI-gestützter Analysen auf den Stand von **März 2026** optimiert worden. Sie ist maßgeschneidert für moderne Hardware-Plattformen wie **AMD Ryzen (AM5)** und **Radeon RX 7000er** GPUs.

Das Ziel ist ein performantes **Btrfs-System**, das mittels **EFISTUB** ohne klassischen Bootloader direkt vom Mainboard gestartet wird. Um die Handhabung so wartungsfreundlich wie möglich zu gestalten, verzichten wir bewusst auf komplexe Subvolumes. Wir nutzen Btrfs in seiner puristischen Form – das vereinfacht die Verwaltung massiv, ohne auf moderne Dateisystem-Vorteile zu verzichten.

Im weiteren Verlauf richten wir die **Xfce-Oberfläche** mit **LightDM** und Autologin ein. Nach dem ersten Systemstart widmen wir uns der optischen Anpassung und der Installation des Paketmanagers **Pamac**. Anstelle einer herkömmlichen Swap-Partition setzen wir auf **zRAM**, welcher die enorme Geschwindigkeit von DDR5-Speicher nutzt, um Daten komprimiert und nahezu latenzfrei im RAM auszulagern. Die Details hierzu findest du im ergänzenden Kapitel „Arch Linux Post-Installation“.

Quellen & weiterführende Links:

- zur Arch Linux Post-Installation
- zur Videoreihe: Total Arch Linux (unicks.eu)
- Arch Linux Wiki – Anleitung für Einsteiger

... zur Übersicht



b. Setup-Checkliste: Voraussetzungen für das Arch-System

Bevor wir mit der Installation beginnen, definieren wir das technische Fundament. Diese Anleitung ist primär auf ein modernes AMD AM5-System zugeschnitten, die Konzepte lassen sich jedoch problemlos auf jede aktuelle Hardware-Generation übertragen, die auf UEFI und NVMe-Speicher setzt.

Ziel ist ein performantes, wartungsarmes Arch-System, das konsequent auf Altlasten verzichtet. Wir setzen auf ein puristisches Setup, das maximale Kontrolle bietet, ohne die Komplexität unnötig zu steigern. Nutze die folgende Liste als Checkliste, um sicherzustellen, dass dein Vorhaben mit diesem Leitfaden übereinstimmt.

Die Spezifikationen

- **Plattform:** Optimiert für AMD AM5 (CPU & GPU), aber allgemeingültig für moderne x86-Systeme.
- **Mainboard-Modus:** Reines **UEFI** (CSM/Legacy zwingend deaktiviert).
- **Sicherheit:** Secure Boot ist für die Installation **deaktiviert**.
- **Datenträger:** NVMe SSD (beispielhaft 4 TB) mit **GPT-Partitionstabelle** (via gdisk).
- **Betriebssystem:** Arch Linux 64-bit als **Single-Boot** (alleiniges OS).
- **Dateisystem:** **Btrfs** auf der gesamten Root-Partition (ohne Subvolumes).
- **Bootloader:** **EFISTUB** – direkter Boot durch das UEFI (kein GRUB/systemd-boot erforderlich).
- **Desktop-Umgebung:** **Xfce**.
- **Netzwerk:** LAN-Verbindung für die Installation; danach Verwaltung via **NetworkManager**.
- **Software-Basis:** Aktiviertes **Multilib-Repository** (für Steam/Gaming/32-Bit).
- **Besonderheit:** Keine Verschlüsselung; optimiert für ein anschließendes zRAM-Setup.

... zur Übersicht



c. BIOS/UEFI Vorbereitung für aktuelle AM5 Motherboards

Bevor wir mit der Installation von Arch Linux beginnen, muss das System auf einen stabilen und performanten Stand gebracht werden. Die folgenden Einstellungen sind primär auf **MSI-Mainboards** (B650, X670) mit Ryzen 7000er/8000er CPUs zugeschnitten, lassen sich aber dank der einheitlichen AM5-Logik auf alle Hersteller übertragen.

Die Logik hinter diesen Einstellungen ist bei allen AM5-Herstellern identisch, auch wenn die Menübezeichnungen variieren. Während MSI von M-Flash spricht, heißt die Update-Funktion bei ASUS EZ Flash und bei Gigabyte Q-Flash. Das RAM-Profil EXPO ist der herstellerübergreifende Standard für AMD-Systeme. Am **Ende des Kapitels** werden auch die BIOS-Einstellungen und auch die BIOS-Pfade für **ASUS AM5** und **Gigabyte AM5** aufgelistet.

Schnellzugriff: BIOS & Boot-Menü

Nutze diese Tasten beim Systemstart, um in die Konfiguration oder die Laufwerksauswahl zu gelangen:

Hersteller	BIOS-Setup (Einstellungen)	Boot-Menü (Schnellauswahl)
MSI	Entf (Del)	F11
ASUS	Entf (Del) oder F2	F8
Gigabyte	Entf (Del)	F12
ASRock	Entf (Del) oder F2	F11

BIOS-Optimierungen (Fokus: MSI Click BIOS 5)

1. BIOS-Update (M-Flash):

Ein aktuelles BIOS ist unter AM5 essenziell für die Systemstabilität und die Initialisierung von NVMe-Laufwerken.

- Lade das neueste **stabile** BIOS von der MSI Website.
- Entpacke die Datei auf einen **FAT32-formatierten USB-Stick**. (Datei nicht umbenennen!)
- Nutze idealerweise einen **USB-2.0-Port** an der Rückseite des Boards für maximale Stabilität.
- Starte ins BIOS (Entf) und wähle **M-Flash**. Der PC startet neu und führt das Update durch.

Wichtig: Schalte den PC während des Vorgangs niemals aus!

Hinweis: Formatiere den USB-Stick auf FAT32. Das ist das einzige Format, das jedes BIOS zuverlässig erkennt.

2. RAM-Stabilität & Boot-Präzision

Um die volle Geschwindigkeit deines Speichers zu nutzen, ohne System-Freezes zu riskieren:

- **Schritt 1 (EXPO):** Aktiviere **Profile 1**, um den RAM mit seinem Ziel-Takt (z. B. 6000 MHz) zu betreiben.

- **Schritt 2 (Stabilitäts-Check):** Lasse **Memory Context Restore** und **Power Down Enable** auf **Disabled** (oder Auto).
- Warum? Bei hohen Taktraten wie 6000 MHz sorgt das RAM-Training bei jedem Start (ca. 40 Sek.) dafür, dass die Spannungen perfekt kalibriert werden. Dies verhindert die berüchtigten Desktop-Freezes der AM5-Plattform.

3. CPU-Performance (PBO & Curve Optimizer)

Vermeide den "MSI Game Boost", da dieser oft mit unnötig hohen Spannungen arbeitet. Nutze stattdessen den effizienten Weg:

- **PBO (Precision Boost Overdrive):** Auf **Advanced** stellen.
- **Curve Optimizer:** All Cores / Sign: **Negative** / Magnitude: **10** bis **20**.
- Effekt: Die CPU verbraucht weniger Strom, bleibt kühler und hält den Boost-Takt länger stabil. Der Wert **10** ist ein extrem sicherer "Sweet Spot".

4. Linux-Spezifische Fixes (Idle & Sleep)

Um zufällige Hänger im Leerlauf oder beim Aufwachen zu verhindern:

- **Global C-state Control:** Auf **Enabled** lassen (Wichtig für das Stromsparen unter Linux), aber kombiniert mit:
- **Power Supply Idle Control:** Auf **Typical Current Idle** stellen.
- Warum? Dies stabilisiert die Stromzufuhr der CPU im Leerlauf und verhindert "Silent Freezes", wenn das System nichts zu tun hat.

5. Boot-Konfiguration für das Arch-Medium

- **Boot Mode:** Zwingend auf **UEFI** (CSM/Legacy deaktivieren).
- **Full Logo Display:** Auf **Enabled** stellen. Dies stellt sicher, dass das MSI-Logo beim Starten im Grafikspeicher (GOP) bereitgestellt wird, damit Linux es später für den „Silent Boot“ übernehmen kann.
- **SATA Mode:** **AHCI** (Standard).
- **Secure Boot:** Auf **Disabled** (Da der Arch-Kernel standardmäßig nicht signiert ist).

🚩 **6. Vom BIOS zum Live-System** Speichere alle Einstellungen mit F10. Wähle beim Neustart über das **Boot-Menü** deinen USB-Stick aus. Du bist nun bereit für die Installation.

Vergleichbare BIOS-Pfade für AM5 (Die "Großen Vier")

Solltest du kein MSI-Board nutzen, findest du hier die entsprechenden Pfade für ASUS, Gigabyte und ASRock:

Funktion	MSI (Click BIOS)	ASUS (Ai Tweaker)	Gigabyte (Tweaker)	ASRock (OC Tweaker)
Update	M-Flash	Tool → EZ Flash 3	F8 oder Q-Flash	Tool → Instant Flash
RAM	OC → EXPO	Ai Overclock → EXPO	Extreme Memory Prof.	OC Tweaker → EXPO

Funktion	MSI (Click BIOS)	ASUS (Ai Tweaker)	Gigabyte (Tweaker)	ASRock (OC Tweaker)
MCR	OC → MCR	DRAM → MCR	DDR Training → MCR	DRAM Config → MCR
PBO	Advanced → AMD OC	Advanced → AMD OC	Settings → AMD OC	Advanced → AMD OC
Curve Opt.	Im PBO-Menü	Im PBO-Menü	Im PBO-Menü	Im PBO-Menü
C-States	Advanced → C-state	AMD CBS → C-state	AMD CBS → C-state	Advanced → C-state
Secure Boot	Settings → Security	Boot → OS Type: Other	Boot → Secure Boot: Off	Security → Secure Boot

Wichtige Hinweise zur Tabelle:

- **ASUS Secure Boot:** Die Einstellung "**Other OS**" entspricht bei ASUS dem Deaktivieren des Secure Boot für Nicht-Windows-Systeme wie Arch Linux.
- **Kombi-Fix:** Aktiviere zusammen mit **Memory Context Restore (MCR)** immer auch **Power Down Enable**, um Instabilitäten zu vermeiden.

Hersteller-Support & BIOS-Downloads (AM5)

Um das korrekte BIOS für dein spezifisches Board-Modell zu finden, nutze die folgenden offiziellen Portale. **Wichtig:** Achte bei der Auswahl auf die exakte Modellbezeichnung und ggf. die Revisionsnummer (z. B. Rev. 1.0 vs. 1.1), die meist unten links auf dem Mainboard aufgedruckt ist.

Hersteller	Webseite
MSI	MSI Support Center
MSI speziell für:	MSI Carbon Wifi B650
ASUS	ASUS Download Center
Gigabyte	Gigabyte Support
ASRock	ASRock Support-Suche

⚠ **Was tun, wenn der PC nicht startet? (BIOS-Reset)**

Nach RAM-Änderungen kann der erste Start bis zu 3 Minuten dauern ("Memory Training").

>Bleibt der Bildschirm länger schwarz:

1. PC vom Strom trennen.
2. Die **CMOS-Batterie** für 30 Sek. entfernen oder die Pins **JBAT1** (MSI) / **CLR_CMOS** kurzschließen.
3. Das BIOS ist nun auf Werkszustand zurückgesetzt.

[... zur Übersicht](#)



d. Das Installations-Medium: Arch Linux startklar machen

Bevor wir dein MSI AM5-System mit Arch Linux zum Leben erwecken, benötigen wir ein absolut zuverlässiges **Installations-Medium**. Da moderne UEFI-Mainboards sehr präzise Anforderungen an den Boot-Prozess stellen, ist die korrekte Erstellung des USB-Sticks das A und O. In diesem Kapitel erfährst du, wie du das offizielle ISO-Abbild sicher herunterlädst, verifizierst und fehlerfrei auf deinen Stick schreibst.

1. Download & Prüfung

↓ <https://www.archlinux.de/download/>

Um sicherzugehen, dass der Download fehlerfrei war, vergleichen wir die SHA256-Prüfsumme.

So führst du die Prüfung durch:

Damit das Terminal die Datei findet, musst du den Befehl in dem Ordner ausführen, in dem dein Download liegt (meistens der Ordner Downloads).



Tip: Klicke in deinem Dateimanager mit der rechten Maustaste in den leeren Bereich des Download-Ordners und wähle "**Im Terminal öffnen**".

Gib nun den Anfang des Befehls ein und nutze die **Tab-Taste** (**↵**) zur automatischen Ergänzung:

```
sha256sum arch + ↵
```

Das Terminal vervollständigt den Namen automatisch zu:

```
sha256sum archlinux-2025.03.01-x86_64.iso
```

- **2025.03.01:** Steht für das **Release-Datum** (Jahr.Monat.Tag). Bei dir werden dort die Zahlen deiner aktuell geladenen Version stehen.
- **x86_64:** Bezeichnet die **Architektur** (64-Bit), die für alle modernen PCs und AM5-Systeme Standard ist.
- **.iso:** Das Dateiformat des CD/DVD-Abbilds.

Checkliste: So prüfst du das ISO richtig

- **Korrekte Datei?** Vergleiche den Dateinamen deines Downloads genau mit der Angabe auf der Webseite (z. B. archlinux-2025.03.01-x86_64.iso).
- **Der Hash-Typ:** Achte darauf, ob die Webseite **MD5**, **SHA1** oder **SHA256** angibt. Nutze den entsprechenden Befehl (z. B. sha256sum), falls MD5 nicht gelistet ist.
- **Zeichen-Abgleich:** Du musst nicht jede Ziffer einzeln lesen. Vergleiche einfach die **ersten 4** und die **letzten 4** Zeichen der Ausgabe mit der Angabe auf der Seite. Stimmen diese überein, ist die Datei zu 99,9 % korrekt.
- **Fehlerfall:** Weicht auch nur ein einzelnes Zeichen ab? **Lösche die ISO** und lade sie neu herunter. Ein beschädigtes Image führt später zu kryptischen Fehlern bei der Installation der Pakete.

2. Identifikation des USB-Sticks



Stecke den Stick ein und ermittle den Laufwerksbuchstaben (z. B. sdb) mit:

```
lsblk
```

Sollte der Stick Partitionen enthalten (z. B. sdb1, sdb2), müssen diese zwingend ausgehängt werden, damit der Schreibvorgang nicht blockiert wird. Der Stern (*) am Ende sorgt dafür, dass alle Partitionen des Sticks gleichzeitig gelöst werden:

```
sudo umount /dev/sdx*
```

Ersetze das x durch deinen ermittelten Buchstaben, z. B. sdb*

-  **Tipp:** Falls umount meldet, dass das Ziel nicht eingehängt ist, kannst du diesen Schritt ignorieren und direkt mit dem Schreiben fortfahren.
-  **Achtung:** Gib das gesamte Laufwerk an (z. B. /dev/sdb), **niemals** eine Partition (wie /dev/sdb1). Alle Daten auf dem Stick werden gelöscht!

3. Erstellungsmethoden


Hier stehen dir drei bewährte Wege zur Verfügung:

Methode A: Das Terminal (dd-Befehl)

Dies ist die zuverlässigste Methode unter Linux. Das Image wird bitgenau geschrieben:

```
sudo dd if=/pfad/zu/archlinux-2025.03.01-x86_64.iso of=/dev/sdx bs=4M status=progress oflag=sync
```

- **Hinweis:** 2025.03.01: Steht für das **Release-Datum** (Jahr.Monat.Tag). Bei dir werden dort die Zahlen deiner aktuell geladenen Version stehen.

 Ausführliche Anleitung: Linux Bootstick erstellen

Methode B: Grafisch (Gnome-Disk-Utility)

Ideal für Nutzer, die eine GUI bevorzugen ("Laufwerke"):

1. USB-Stick in der Liste auswählen.
2. Oben rechts auf das Menü (drei Punkte) klicken.
3. **"Laufwerksabbild wiederherstellen..."** wählen und die ISO-Datei selektieren.

Methode C: Ventoy (Multiboot)

Wenn du bereits einen Ventoy-Stick nutzt, kopiere die ISO-Datei einfach per Drag-and-Drop in das Hauptverzeichnis des Sticks.

 Anleitung zu Ventoy Multibootstick

... zur Übersicht



e. Willkommen in der Konsole: Die Arbeitsumgebung vorbereiten

Der erste Meilenstein ist geschafft: Das Live-System bootet! Bevor wir uns an die eigentliche Installation auf der NVMe-SSD machen, richten wir uns in der Konsole häuslich ein. Da die Standardumgebung von Arch Linux sehr puristisch startet, sorgen wir in diesem Schritt für ein deutsches Tastaturlayout und eine gut lesbare Schriftdarstellung. So behältst du bei den kommenden Befehlen die volle Kontrolle und vermeidest unnötige Tippfehler.

a. Der Boot-Vorgang

1. Stecke deinen vorbereiteten **Arch Linux USB-Stick** in einen der hinteren USB-Ports (direkt am Mainboard, nicht am Gehäuse-Frontpanel, um Übertragungsfehler zu vermeiden).
2. Starte den PC neu und drücke mehrfach die Taste **F11** (bei MSI), um das **Boot-Menü** zu öffnen.
3. Wähle den Eintrag aus, der mit **UEFI:** beginnt (z. B. UEFI: USB Partition 1).

Nach dem Bootvorgang landest du in einer minimalistischen Kommandozeile als Root-Benutzer:

```
root@archiso ~ #
```

b. Tastaturlayout & Anzeige optimieren

Standardmäßig nutzt die Konsole das US-Layout. Um Sonderzeichen wie den Bindestrich korrekt zu finden, stellen wir auf Deutsch um.

Schritt 1:

Gib zuerst folgenden Befehl ein, um das grundsätzliche deutsche Layout zu laden. **Hinweis:** Da das US-Layout noch aktiv ist, drücke für das **y** die Taste **z**.

```
loadkeys de
```

Schritt 2:

Nun, da der Bindestrich (-) und die Tasten y/z bereits an der gewohnten Stelle liegen, optimieren wir das Layout auf den Standard:

```
loadkeys de-latin1
```

Für eine deutlich bessere Lesbarkeit auf hochauflösenden Monitoren (4K/WQHD) empfiehlt es sich zudem, eine größere Schriftart zu laden. Mein Favorit für AM5-Systeme mit modernen Displays:

Schritt 3:

```
setfont ter-132n
```

Sollte ter-132n nicht im Standard-ISO von Arch enthalten sein, gibt es noch weitere Alternative für große Schriften:

```
setfont latarcyrheb-sun32
setfont ter-v32b
```

Wusstest du schon?

Der Befehl **loadkeys de** schaltet sofort auf das deutsche QWERTZ-Layout um (Y/Z-Tausch fixen). Der Zusatz **-latin1** stellt sicher, dass alle westeuropäischen Sonderzeichen (wie @, | oder ~) exakt so kodiert werden, wie sie auf deiner Tastatur aufgedruckt sind. Das ist besonders wichtig, wenn wir später Konfigurationsdateien im Editor bearbeiten.

c. Nützliche Helfer für die Konsole

- **Num-Block:** Vergiss nicht, die **NUM-Taste** auf deinem Ziffernblock zu aktivieren, falls du Zahlen darüber eingeben möchtest.
- **Übersicht:** Wenn der Bildschirm mit Text überladen ist, sorgt dieser Befehl für eine leere, saubere Konsole: **clear**

... zur Übersicht

f. Verbindung steht? Netzwerk- und Zeitabgleich

Da Arch Linux ein Rolling-Release-System ist, werden alle Pakete während der Installation in ihrer aktuellsten Version direkt von den offiziellen Servern geladen. Eine aktive und stabile Internetverbindung ist daher die Grundvoraussetzung für den gesamten weiteren Prozess.

In dieser Anleitung nutzen wir eine kabelgebundene LAN-Verbindung, da diese vom Arch-Live-System in der Regel automatisch erkannt und ohne manuelle Konfiguration per DHCP initialisiert wird. Bevor wir mit der Partitionierung der Festplatte beginnen, stellen wir sicher, dass die Kommunikation mit der Außenwelt steht und die Systemzeit für eine fehlerfreie Verschlüsselung (HTTPS) synchronisiert ist.

1. Der Verbindungstest

Da wir ein LAN-Kabel verwenden, sollte die Internetverbindung automatisch beim Booten hergestellt worden sein. Wir prüfen dies mit einem kurzen Ping-Befehl:

```
ping -c 3 google.de
```

Erhältst du eine Antwort (64 bytes from...), steht die Verbindung. Falls nicht, prüfe das Kabel oder starte den Netzwerkdienst manuell mit:

```
systemctl start systemd-networkd
```

2. Zeit-Synchronisation (Wichtig für HTTPS)

Zusätzlich müssen wir die Systemzeit synchronisieren. Dies ist zwingend erforderlich, damit verschlüsselte Verbindungen (HTTPS) bei den späteren Downloads korrekt funktionieren und die Paket-Signaturen als gültig erkannt werden:

```
timedatectl set-ntp true
```

Bei Bedarf kannst Du kontrollieren, ob die Zeit erfolgreich abgeglichen wurde (Achte auf "System clock synchronized: yes"):

```
timedatectl status
```

... zur Übersicht



g. Das Fundament: Partitionierung einer NVMe SSD

Bevor wir Arch Linux auf deiner SSD installieren können, müssen wir den Datenträger strukturieren. Man kann sich diesen Vorgang wie das Einziehen von Zwischenwänden in ein großes, leeres Gebäude vorstellen: Wir unterteilen den Speicherplatz in feste Bereiche, die sogenannten **Partitionen**.

Da moderne Computer heute den **UEFI-Standard** nutzen, setzen wir auf das aktuelle **GPT-Layout** (GUID Partition Table). Dies stellt sicher, dass dein Mainboard das Betriebssystem beim Start sofort erkennt und die volle Kapazität moderner Datenträger (wie deiner 4 TB SSD) problemlos nutzen kann. Unser Ziel ist ein minimalistisches „Zwei-Bereich-Layout“:

1. Eine kleine **Boot-Partition** für den Startvorgang des Kernels.
2. Eine große **Root-Partition** für das gesamte Betriebssystem und deine Daten.

Wir verwenden hierfür das Werkzeug **gdisk**, da es Partitionen auf modernen Speichermedien automatisch präzise ausrichtet (Alignment). Das schont die Hardware und sorgt dafür, dass deine SSD ihre volle Geschwindigkeit von Anfang an entfalten kann.

⚠ **Wichtiger Sicherheitshinweis:**

Verschafe dir mit **lsblk** zuerst einen genauen Überblick über deine Laufwerke. Achte penibel darauf, die richtige SSD (z. B. `/dev/nvme0n1`) zu adressieren, da bei diesem Vorgang alle vorhandenen Daten gelöscht werden!

```
lsblk
```

1. **gdisk** starten und Initialisieren

Starte das Partitionierungswerkzeug für deine SSD (ersetze den Pfad, falls deine SSD anders benannt ist, z. B. `/dev/nvme0n1`):

```
gdisk /dev/nvme0n1
```

Da wir Arch Linux als alleiniges Betriebssystem auf dieser SSD installieren, erstellen wir als Erstes eine komplett neue, leere **GPT-Partitionstabelle**. Dies löscht alle eventuell vorhandenen Strukturen und sorgt für ein sauberes Fundament:

- Tippe **o** (für eine neue GPT-Tabelle).
- Bestätige die Sicherheitsabfrage mit **Y**.

Nun ist die SSD bereit für unser minimalistisches Zwei-Partitionen-Layout.

2. Das Partitionsschema erstellen:

A. Die BOOT-Partition (/dev/nvme0n1p1) – für Kernel & EFISTUB

Aktion (Terminal-Abfrage)	Eingabe 1	Eingabe 2
Command	n	ENTER
Partition number		ENTER
First sector		ENTER
Last sector	+1024M	ENTER
Hex code	ef00	ENTER

B. Die ROOT-Partition (/dev/nvme0n1p2) – für das Arch Linux System

Aktion (Terminal-Abfrage)	Eingabe 1	Eingabe 2
Command	n	ENTER
Partition number		ENTER
First sector		ENTER
Last sector		ENTER
Hex code	8300	ENTER

⚠ Wichtige Hinweise zum Partitionierungsvorgang:

- **Restlicher Speicherplatz:** Wird bei der zweiten Partition (nvme0n1p2) bei "Last sector" kein Wert eingetragen (einfach ENTER), weist gdisk automatisch den gesamten verbleibenden Speicherplatz der SSD dieser Partition zu.
- **Übersicht behalten:** Die Eingabe von **p** zeigt dir jederzeit den aktuellen Stand deiner Partitionstabelle an, bevor du sie endgültig speicherst.
- **Änderungen übernehmen:** Erst durch die Eingabe von **w** (Write) und der anschließenden Bestätigung mit **Y** werden die Partitionen tatsächlich auf die SSD geschrieben. Bis zu diesem Moment wurden keine Daten verändert.
- **Erfolgsmeldung:** Nach dem Speichern quittiert gdisk den Vorgang mit: "OK, writing new GUID partition table (GPT) to /dev/nvme0n1. The operation has completed successfully." Ein abschließendes **lsblk** im Terminal zeigt dir nun deine neue Struktur.
- **Initialisierung:** Falls die SSD bereits eine Partitionstabelle enthielt, fragt gdisk nach dem Befehl **o** sicherheitshalber nach einer Bestätigung. Tippe hier **Y**, um die alte Tabelle unwiderruflich zu überschreiben.

... zur Übersicht



h. Dateisysteme: Btrfs-Optimierung und Mount-Struktur

Nachdem die physische Aufteilung der SSD abgeschlossen ist, müssen wir die Partitionen „formatieren“, damit das Betriebssystem Daten darauf speichern kann. Während wir für die Boot-Schnittstelle das universelle **FAT32** nutzen, kommt für das Hauptsystem das moderne **Btrfs** zum Einsatz.

Dieses Dateisystem ist besonders für leistungsstarke NVMe-SSDs prädestiniert: Durch gezielte Mount-Optionen wie die **Zstd-Kompression** reduzieren wir nicht nur den Platzbedarf, sondern minimieren auch die Schreiblast, was die Lebensdauer deiner SSD verlängert. Gleichzeitig sorgt das **asynchrone Discard** dafür, dass dein System auch bei intensiven Löschvorgängen reaktionsschnell bleibt. Wir verzichten hierbei bewusst auf komplexe Subvolumes, um die Pfadstruktur für dich so einfach und wartungsfreundlich wie möglich zu halten.

1. Dateisysteme anlegen (Formatieren)

Mit dem Befehl **mkfs** (make file system) werden die Partitionen formatiert.

Hinweis: Bei der EFI-Partition muss das Label (Name) zwingend in Großbuchstaben geschrieben werden.

Boot-Partition (EFI):

```
mkfs.fat -F32 -n BOOT /dev/nvme0n1p1
```

Root-Partition (Btrfs):

```
mkfs.btrfs -L ROOT /dev/nvme0n1p2
```

2. Partitionen einhängen (Mounten)

Damit wir Arch Linux installieren können, müssen die Partitionen in das Live-System eingebunden werden. Wir nutzen dabei moderne Mount-Optionen, um die Performance deiner NVMe-SSD zu maximieren:

a. System-Partition (ROOT) einhängen:

```
 mount -o compress=zstd:3,ssd,discard=async /dev/nvme0n1p2 /mnt
```

- **compress=zstd:3:** Komprimiert Daten im Hintergrund, spart Platz und erhöht die Lebensdauer der SSD.
- **discard=async:** Sendet TRIM-Befehle asynchron, was Ruckler bei großen Löschvorgängen verhindert.

b. Boot-Partition einhängen:

Zuerst erstellen wir den Einhängpunkt innerhalb des bereits gemounteten ROOT-Systems und binden dann die EFI-Partition ein:

```
mkdir /mnt/boot
mount /dev/nvme0n1p1 /mnt/boot
```

3. Kontrolle und weitere Datenträger

Nach dem Formatieren kannst du die neuen Unique-IDs (UUIDs) abrufen, falls du sie später für Konfigurationen manuell benötigst:

```
blkid /dev/nvme0n1p1
blkid /dev/nvme0n1p2
```

Optionale Daten-Partition:

Möchtest du eine weitere interne Festplatte (z. B. /dev/sda1) direkt in das neue System einbinden, erstelle auch hierfür einen Ordner und mounte sie:

```
mkdir /mnt/datenplatte
mount /dev/sda1 /mnt/datenplatte
```

Hinweis: Alle hier manuell vorgenommenen Mounts werden später durch den Befehl **genfstab** automatisch in die Konfiguration nach /etc/fstab übernommen, damit sie bei jedem Systemstart automatisch verfügbar sind.

... zur Übersicht



i. Server-Tuning: Schnelle Spiegelserver für den Download

Bevor wir das eigentliche Betriebssystem herunterladen, optimieren wir unsere „Versorgungswege“. Arch Linux wird weltweit auf hunderten Servern – sogenannten **Mirrors** – zum Download bereitgestellt. Standardmäßig greift das System auf eine bunte Mischung internationaler Server zu, was die Installation unnötig in die Länge ziehen kann.

Stell dir vor, du bestellst Pakete: Es macht einen riesigen Unterschied, ob der Bote aus der Nachbarstadt kommt oder erst einen Ozean überqueren muss. Mit dem Werkzeug **Reflector** filtern wir gezielt die schnellsten und aktuellsten Server in deiner Nähe (Deutschland) heraus. So stellen wir sicher, dass deine Internetleitung voll ausgelastet wird und die Installation der Basis-Pakete in Bestzeit erledigt ist.

1. Automatische Optimierung mit Reflector

Zuerst gleichen wir die Systemuhr mit dem Internet ab, um SSL Fehler zu verhindern:

```
timedatectl set-ntp true
```

Mit dem Reflectorbefehl können wir nun die schnellsten und aktuellsten HTTPS-Server aus Deutschland finden.

```
 reflector --country Germany --latest 20 --protocol https --sort rate --save /etc/pacman.d/mirrorlist
```

Hinweis: Dieser Vorgang dauert einen Moment (ca. 10–30 Sek.). Solange der Cursor blinkt, testet das System im Hintergrund die Verbindungsgeschwindigkeiten. Einzelne Fehlermeldungen über nicht erreichbare Server können dabei ignoriert werden.

⚠ **Wichtig:** Alle Parameter beginnen mit zwei Bindestrichen (z. B. --country). Achte darauf, keine Leerzeichen zwischen die Striche zu setzen.

Erklärung der Befehle & Parameter:

- **timedatectl set-ntp true:** Ohne korrekte Uhrzeit schlägt die Zertifikatsprüfung der HTTPS-Server fehl. Reflector würde dann gar keine Server finden („No mirrors found“).
- **--country Germany:** Beschränkt die Suche auf Server in deiner Nähe für niedrige Latenzen.
- **--latest 20:** Begrenzt die Suche auf die 20 aktuellsten Server.
- **--protocol https:** Wir erzwingen verschlüsselte Verbindungen für maximale Sicherheit bei den Paketen.
- **--sort rate:** Sortiert die Liste so, dass die schnellsten Server für deinen Anschluss ganz oben stehen.

2. Manuelle Kontrolle (Optional)

Falls du die Liste einsehen oder manuell Server verschieben möchtest, kannst du die Datei mit dem Texteditor **nano** öffnen:

```
nano /etc/pacman.d/mirrorlist
```

Nützliche Nano-Befehle:

- **STRG + K**: Löscht die aktuelle Zeile (schneidet sie aus).
- **STRG + U**: Fügt die ausgeschnittene Zeile an der Cursor-Position wieder ein.
- **STRG + O** gefolgt von **ENTER**: Speichert die Änderungen.
- **STRG + X**: Beendet den Editor.

Anmerkung: Die hier vorgenommene Optimierung gilt nur für die aktuelle Installations-Sitzung. Später werden wir in Kapitel "o. Optimierung der Paketverwaltung" die dauerhafte Mirrorlist-Optimierung für dein fertiges System einrichten.


... zur Übersicht

j. Das Rohbausystem | Installation der Basis-Pakete mit Pacstrap

Nachdem das Fundament steht und die Partitionen vorbereitet sind, folgt nun der eigentliche Aufbau des Systems. In diesem Schritt übertragen wir die essenziellen „Bausteine“ des Betriebssystems von unserem USB-Stick auf die SSD.

Mit dem Werkzeug **pacstrap** wählen wir ganz gezielt die Komponenten aus, die perfekt zu deiner Hardware passen. Wir installieren den Linux-Kernel als Herzstück des Systems, die notwendigen Mikrocode-Updates für deine AMD-CPU sowie das Grafik-Fundament für deine Radeon-Karte. Damit stellen wir sicher, dass das System von Beginn an optimal auf die verbauten Komponenten abgestimmt ist und alle grundlegenden Funktionen für den späteren Betrieb bereitstehen.

Der Installationsbefehl:

```
 pacstrap -K /mnt base base-devel linux linux-firmware amd-ucode networkmanager btrfs-progs nano efibootmgr reflector
```

Erläuterungen zu den gewählten Paketen:

- **base & base-devel:** "base" bildet das minimale Grundgerüst des Systems. "base-devel" enthält zusätzliche Werkzeuge (wie gcc oder make), die später zwingend erforderlich sind, um Software aus dem **AUR** (Arch User Repository) zu bauen.
- **-K:** Dieser Parameter sorgt dafür, dass die digitalen Schlüssel (Keyrings) für die Paketprüfung bereits jetzt korrekt vom Live-System in das neue System kopiert werden.
- **amd-ucode:** Enthält wichtige Mikrocode-Updates für deine AMD-CPU, die Fehler korrigieren und die Stabilität verbessern. Da wir ein modernes System nutzen, ist dieses Paket **zwingend erforderlich**.
- **networkmanager:** Wir installieren den Dienst bereits jetzt mit, damit du nach dem ersten Neustart sofort ein mächtiges Werkzeug zur Internetkonfiguration (z. B. via nmcli oder später in der GUI) zur Verfügung hast.
- **btrfs-progs:** Stellt die notwendigen Dienstprogramme für dein Btrfs-Dateisystem bereit.
- **Grafik-Fundament (amdgpu & mesa):** Diese Pakete bilden das Fundament für deine AMD-Grafikkarte. Damit ist sichergestellt, dass deine Xfce-Oberfläche später sofort mit voller Hardwarebeschleunigung startet.
- **efibootmgr:** Dies ist unser wichtigstes Werkzeug für das spätere Boot-Management. Es erlaubt uns, den Arch-Kernel ohne Umwege direkt im UEFI deines Mainboards als Startoption zu registrieren (**EFISTUB**).

... zur Übersicht

k. **fstab**: Laufwerke dauerhaft verankern

Bisher haben wir die Partitionen deiner SSD manuell in das Live-System eingehängt. Damit dein neues Arch Linux aber nach einem Neustart ganz von alleine weiß, welche Partitionen an welchen Stellen im Dateisystem eingehängt werden sollen, benötigen wir die Konfigurationsdatei **fstab** (file system table). Man kann sie sich wie eine dauerhafte „Wegbeschreibung“ oder einen Dienstplan vorstellen, den das System bei jedem Start ausliest.

Wir generieren diese Datei automatisch auf Basis deiner aktuellen Mount-Punkte. Dabei nutzen wir **UUIDs** (Universally Unique Identifier). Diese eindeutigen Hardware-Identifikationsnummern stellen sicher, dass dein System auch dann korrekt bootet, wenn sich die Laufwerksreihenfolge – etwa durch das Anschließen weiterer USB-Sticks oder SSDs – nachträglich ändert. Gleichzeitig „verankern“ wir hier wichtige Btrfs-Optimierungen, damit deine NVMe-SSD von der ersten Sekunde an mit maximaler Performance und Effizienz arbeitet.

1. Die **fstab** automatisch generieren

Wir nutzen das Werkzeug **genfstab**, um die aktuelle Mount-Struktur in die Konfigurationsdatei zu schreiben.

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


- **-U**: Verwendet die eindeutigen Hardware-IDs (UUIDs) statt der Gerätenamen wie `/dev/nvme0n1p2`. Dies ist für einen stabilen Bootvorgang mit EFISTUB essenziell.

2. Mount-Optionen optimieren (Btrfs)

Nach der Erzeugung prüfen und ergänzen wir die Datei, um die volle Performance und Hardware-Schonung deiner NVMe-SSD zu aktivieren:

```
nano /mnt/etc/fstab
```

Suche die Zeile für deine Root-Partition (`/`) und stelle sicher, dass die Btrfs-Optionen wie folgt hinterlegt sind (oder ergänze sie händisch):

```
 rw,relatime,compress=zstd:3,ssd,discard=async
```

und dort die Btrfs-Optionen entsprechend ergänzen mit:

⚠ **Wichtige Syntax-Regel**: Achte beim Bearbeiten mit nano peinlich genau darauf, dass zwischen den Optionen (z. B. `ssd,compress=zstd:3`) **nur Kommas und keine Leerzeichen** stehen. Ein Leerzeichen an der falschen Stelle kann den Bootvorgang verhindern.

3. Beispiel einer fertigen fstab

Mit dem Befehl `cat /mnt/etc/fstab` kannst du das Ergebnis kontrollieren. Eine optimal konfigurierte Datei für dein System sieht etwa so aus:

Einträge in der fstab: /mnt/etc/fstab



Die System-Partition (Root): # /dev/nvme0n1p2

```
UUID=f8f9fe27-1810-49ba-ae31-032eebe7ff3d / btrfs
rw,relatime,compress=zstd:3,ssd,discard=async,space_cache=v2,subvol=/ 0 0
```

Die Boot-Partition (EFI-Boot): # /dev/nvme0n1p1

```
UUID=EC17-4EF7 /boot vfat
rw,relatime,mask=0022,dmask=0022,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro 0 2
```

Interne Datenträger: # HDD/SSD

```
UUID=4934f036-428a-469a-b3d5-a4ed8ae1fc67 /Daten ext4 rw,relatime,nofail,x-gvfs-show 0 2
```

4. Profi-Tipps für zusätzliche Laufwerke

- **Ausfallsicherheit (nofail):** Die Option **nofail** bei internen Datenträgern ist ein "Lebensretter". Sollte eine Platte einmal nicht angeschlossen sein, bricht das System den Bootvorgang nicht ab, sondern startet normal bis zum Desktop durch.
- **Komfort im Desktop (x-gvfs-show):** Dieser Parameter ist speziell für deine **Xfce-Oberfläche** gedacht. Er sorgt dafür, dass deine Festplatten direkt in der Seitenleiste des Dateimanagers (Thunar) erscheinen.

... zur Übersicht



I. arch-chroot | Der Wechsel in das neue System

Dies ist ein entscheidender Moment der Installation: Mit dem Befehl **arch-chroot** verlassen wir die Umgebung des Live-USB-Sticks und wechseln virtuell in dein neu installiertes System auf der NVMe-SSD. Ab diesem Zeitpunkt wirken alle weiteren Befehle – wie das Anlegen von Benutzern, das Ändern von Passwörtern oder die Konfiguration des Bootloaders – direkt auf dein zukünftiges Betriebssystem.

Der Systemwechsel:

```
arch-chroot /mnt
```

⚠ Wichtige Schritte direkt nach dem Wechsel:

Tastaturlayout korrigieren: Da die chroot-Umgebung standardmäßig auf das amerikanische Layout zurückspringt, musst du als Erstes wieder auf Deutsch umstellen. Nur so liegen Sonderzeichen wie `|`, `\` oder `"` für die kommenden Konfigurationsschritte an der gewohnten Stelle.

Schritt 1:

Gib zuerst folgenden Befehl ein, um das grundsätzliche deutsche Layout zu laden. **Hinweis:** Da das US-Layout noch aktiv ist, drücke für das **y** die Taste **z**.

```
loadkeys de
```

Schritt 2:

Nun, da der Bindestrich (-) und die Tasten y/z bereits an der gewohnten Stelle liegen, optimieren wir das Layout auf den Standard:

```
loadkeys de-latin1
```

Wichtige Hinweise zum Arbeiten im chroot:

- **Geänderte Pfadstruktur:** Ab jetzt beziehen sich alle Pfade direkt auf dein neues System auf der Festplatte. Aus `/mnt/etc/fstab` wird in deiner aktuellen Sitzung also einfach `/etc/fstab`.
- **Direkte Wirkung:** Alle Befehle (wie `passwd`, `systemctl` oder das Installieren weiterer Pakete) verändern nun nicht mehr das USB-Medium, sondern dein dauerhaftes System.
- **Abschluss:** Wenn die gesamte Konfiguration später abgeschlossen ist, verlässt du diese Umgebung mit dem Befehl `exit`. Erst danach können die Partitionen sicher ausgehängt und das System neu gestartet werden.

... zur Übersicht



m. Identität und Sicherheit: Hostname, Benutzer und Root-Rechte

Nachdem das technische Grundgerüst steht, verleihen wir deinem System nun eine eigene Persönlichkeit und legen die Hausregeln fest. In der vernetzten Welt von heute ist es wichtig, dass dein Rechner einen eindeutigen Namen besitzt, unter dem er im Netzwerk angesprochen werden kann.

Gleichzeitig schaffen wir eine klare Sicherheitsstruktur. Unter Linux ist es oberstes Gebot, nicht permanent mit den allmächtigen Administrator-Rechten (**root**) zu arbeiten. Stattdessen richten wir dir ein persönliches Benutzerkonto für den Alltag ein und konfigurieren das Werkzeug **sudo**. Damit erhältst du eine Art „Sicherheitsschlüssel“, mit dem du administrative Aufgaben nur bei Bedarf und nach Passwordeingabe autorisierst. So bleibt dein System vor versehentlichen Fehlern und unbefugten Zugriffen geschützt.

1. Den Rechnernamen (Hostname) festlegen

Zuerst definieren wir, wie dein PC gegenüber anderen Geräten im Netzwerk auftritt. In diesem Beispiel wählen wir den Namen **cat-pc**:

```
echo cat-pc > /etc/hostname
```

Damit die lokale Adressierung reibungslos funktioniert, ergänzen wir diese Informationen in der Datei `/etc/hosts`:

```
nano /etc/hosts
```

Füge dort folgende Zeilen ein:

```
127.0.0.1    localhost
::1         localhost
127.0.1.1    cat-pc
```

2. Passwörter und Benutzerkonten

Zuerst vergeben wir ein sicheres Passwort für den Systemadministrator (root):

```
passwd
```

Nun legen wir deinen persönlichen Benutzer (hier: cat) an und weisen ihm die wichtigsten Hardware-Gruppen zu:

```
useradd -m -G wheel,lp,scanner cat
passwd cat
```

Erläuterungen zu den Gruppen:

- **wheel:** Erlaubt die Nutzung von sudo (Administrator-Rechte).
- **lp:** Ermöglicht den Zugriff auf Drucker.
- **scanner:** Erlaubt die Nutzung der Scan-Funktion von Multifunktionsgeräten.

3. Administrative Rechte zuweisen (sudo)

Damit du später administrative Aufgaben sicher erledigen kannst, ohne dauerhaft als Root-User angemeldet sein zu müssen, schalten wir die sudo-Rechte für die Gruppe wheel frei:

```
EDITOR=nano visudo
```

Suche in der Datei nach der folgenden Zeile und entferne das Kommentarzeichen (#):

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

Speichere mit **Strg + O**, bestätige mit **Enter** und schließe mit **Strg + X**.

... zur Übersicht



n. System-Sprache und Regionaleinstellungen

Damit dein Arch Linux nicht nur die korrekte Sprache ausgibt, sondern auch Umlaute, Zeitformate und Währungssymbole fehlerfrei verarbeitet, konfigurieren wir nun die sogenannten **Locales**. In diesem Schritt legen wir fest, welche Zeichensätze das System generieren soll, um eine durchgängig deutsche Benutzeroberfläche zu ermöglichen.

Zusätzlich stellen wir sicher, dass das deutsche Tastaturlayout sowohl in der reinen Textkonsole als auch später unter der grafischen Oberfläche dauerhaft hinterlegt ist. Dies verhindert, dass du bei späteren Wartungsarbeiten oder beim Login mühsam nach Sonderzeichen suchen musst.

1. Sprache festlegen (Locales)

Zuerst definieren wir die Systemsprache als Deutsch:

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

Nun müssen wir dem System sagen, welche Zeichensätze es tatsächlich generieren soll. Öffne dazu die Konfigurationsdatei **locale.gen**:

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

Suche die folgenden Zeilen und entferne das Kommentarzeichen (#) am Zeilenanfang:

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

Speichere die Datei ab und generiere die Sprachpakete mit:

```
locale-gen
```

2. Tastaturlayout dauerhaft fixieren

Damit deine Tastaturbelegung auch nach dem Neustart erhalten bleibt, erstellen wir die Konfiguration für die Konsole:

```
echo KEYMAP=de-latin1 > /etc/vconsole.conf
```

Wichtiger Zusatz für den Desktop: Damit Z und Y auch später direkt in der grafischen Oberfläche (Xfce/LightDM) an der richtigen Stelle liegen, führen wir diesen Befehl aus:

```
localectl set-x11-keymap de
```

3. Zeitzone und Hardware-Uhr

Wir verknüpfen dein System mit der deutschen Zeitzone:

 In -sf /usr/share/zoneinfo/Europe/Berlin /etc/localtime

Kontrolle: Sobald Arch Linux das erste Mal startet, kannst du mit folgendem Befehl überprüfen, ob alle Einstellungen korrekt übernommen wurden:

```
localectl status
```

[... zur Übersicht](#)



o. Autopilot aktivieren: Paket-Tuning und Mirror-Automatisierung

Nachdem das Grundsystem auf der SSD liegt, passen wir nun die „Schaltzentrale“ für Software-Updates an. In diesem Schritt optimieren wir den Paketmanager Pacman, damit er die volle Bandbreite deiner Internetleitung durch parallele Downloads nutzt und auch 32-Bit-Anwendungen (wichtig für Gaming) unterstützt.

Gleichzeitig richten wir zwei wichtige Hintergrunddienste – sogenannte **Timer** – ein, die dein System dauerhaft in Schuss halten. Der erste sorgt mittels **Reflector** dafür, dass dein PC stets die schnellsten deutschen Download-Server nutzt, ohne dass du die Liste jemals wieder händisch sortieren musst. Der zweite Dienst kümmert sich um die Sauberkeit deiner SSD: Er löscht regelmäßig veraltete Paket-Dateien im Hintergrund, sodass dein Speicherplatz effizient genutzt wird und das System über Jahre hinweg wartungsfrei und schnell bleibt.

1. Pacman optimieren (Speed & Optik)

Wir beschleunigen die Downloads und verschönern die Ausgabe im Terminal:

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

Suche den Abschnitt [options] und passe die Zeilen folgendermaßen an, indem du die Raute (#) vor Color entfernst. Füge darunter ILoveCandy hinzu und aktiviere die parallelen Downloads:

```
[options]
Color
ILoveCandy
ParallelDownloads = 5
```

Wichtig für Gaming: Damit Steam und andere 32-Bit-Anwendungen funktionieren, aktiviere am Ende der Datei das **Multilib-Repository**, indem du bei beiden Zeilen das # entfernst:

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

Speichere mit **STRG + O** und schließe mit **STRG + X**. Aktualisiere danach die Datenbanken:

```
pacman -Syu
```

2. Reflector-Automatisierung (stets schnelle Server)

Damit dein System auch in Zukunft immer die schnellsten deutschen Server nutzt, automatisieren wir den Suchvorgang. Wir konfigurieren dazu die Steuerungsdatei für den Hintergrunddienst:

```
nano /etc/xdg/reflector/reflector.conf
```

Ersetze den Inhalt durch diese optimierten Filter (jeweils eine eigene Zeile):

```
--save /etc/pacman.d/mirrorlist
--protocol https
--country Germany
--age 12
--sort rate
```

Aktiviere nun den **Timer**, der die Liste ab sofort regelmäßig (standardmäßig einmal pro Woche) im Hintergrund aktualisiert:

```
systemctl enable reflector.timer
```

3. Paket-Cache sauber halten

Arch speichert standardmäßig jedes heruntergeladene Paket dauerhaft. Um deine SSD nicht mit „Paketleichen“ zu füllen, automatisieren wir die Reinigung. Wir installieren dazu die **pacman-contrib** und aktivieren den passenden Timer:

```
pacman -S pacman-contrib
systemctl enable paccache.timer
```

Was du damit erreicht hast:

- **Speed:** 5 Pakete werden gleichzeitig geladen statt nur eines.
- **Wartungsfreiheit:** Dein System sucht sich selbstständig die besten Server und löscht veraltete Cache-Dateien (behält nur die letzten 3 Versionen).
- **Gaming-Ready:** Dein System ist bereit für Steam und 32-Bit-Bibliotheken.

... zur Übersicht

p. Desktop-Vorbereitung: System-Dienste | Sound (PipeWire) | Hardware

Nachdem das Grundgerüst steht, installieren wir nun die Software-Ausstattung, die dein Arch Linux in ein voll einsatzfähiges System verwandelt. Wir setzen dabei konsequent auf moderne Standards: Statt des veralteten PulseAudio nutzen wir **PipeWire**, was für geringere Latenzen und eine exzellente Bluetooth-Unterstützung sorgt.

Zusätzlich integrieren wir wichtige Dienste für die EnergiEVERwaltung, das Drucksystem sowie Treiber-Erweiterungen für Mobilgeräte und Windows-Partitionen. Dies stellt sicher, dass Hardware-Erkennung und Datenaustausch reibungslos funktionieren, sobald du deinen neuen Desktop betrittst.

1. Installation der Software-Pakete

Zur besseren Lesbarkeit sind die Pakete in thematische Blöcke unterteilt:

Installation mit pacman
System-Dienste & Druck:
<code>pacman -S acpid avahi nss-mdns cups cups-pdf ghostscript gsfonts mtools udisks2 git</code>
Medien, Scan & Schriften:
<code>pacman -S appstream-glib libunrar libdvdcss simple-scan sane ttf-dejavu ttf-liberation noto-fonts ttf-opensans</code>
Tools & Dateisysteme:
<code>pacman -S ntfs-3g unrar unzip fastfetch xdg-user-dirs bash-completion</code>
Audio-Zentrum (PipeWire):
<code>pacman -S alsa-tools alsa-utils xfce4-pulseaudio-plugin pipewire pipewire-pulse pipewire-alsa pipewire-jack wireplumber pavucontrol</code>
Netzwerk & Bluetooth:
<code>pacman -S network-manager-applet gvfs gvfs-mtp bluez bluez-utils blueman</code>

2. Wichtige Erläuterungen zu den Diensten

- **acpid:** Steuert die Energieverwaltung (z. B. die Reaktion auf den Power-Button).
- **avahi & nss-mdns:** Ermöglichen die automatische Erkennung von Geräten (Drucker/Server) im Netzwerk via Namen (z. B. mein-drucker.local) statt kryptischer IP-Adressen.
- **cups:** Das Standard-Drucksystem für Linux.
- **gvfs / udisks2:** Sorgen dafür, dass USB-Sticks und Smartphones direkt in der Seitenleiste deines Dateimanagers (Thunar) erscheinen.
- **ntfs-3g:** Ermöglicht den vollen Schreibzugriff auf Windows-formatierte Festplatten.
- **BlueZ & Blueman:** Der komplette Bluetooth-Stack inklusive grafischer Verwaltung für die Taskleiste.
- **PipeWire & Wireplumber:** Das moderne Audio-Zentrum. Es ersetzt PulseAudio vollständig und bietet stabilere Verbindungen für Headsets und professionelle Audio-Anwendungen.

3. Besondere Installations-Hinweise

- **Die richtige Auswahl [1]:** Falls pacman dich während der Installation fragt, welcher Anbieter für den pipewire-session-manager genutzt werden soll, wird dir eine Liste angezeigt (z. B. 1) wireplumber 2) pipewire-media-session). Wähle hier immer die **1** für **wireplumber** (oder drücke einfach **ENTER**, da die [1] meist der Standard ist).
- **Konflikte lösen:** Wenn das System fragt, ob **pipewire-pulse** das herkömmliche pulseaudio ersetzen soll, bestätige dies unbedingt mit **Ja (Y)**.
- **Schriftbild:** Die installierten Schriftarten (Noto, DejaVu) garantieren, dass Webseiten und Menüs von Anfang an sauber und ohne "Kästchen" dargestellt werden.

... zur Übersicht



q. Grafik-Power: X-Server, AMD-Treiber und Early KMS

In diesem Kapitel erwecken wir deine Hardware zum Leben. Wir installieren das Anzeige-System und die spezialisierten Treiber für deine AMD Radeon Grafikkarte. Da AMD-Treiber quelloffen direkt im Linux-Kernel integriert sind, profitierst du von maximaler Stabilität und einer hervorragenden Gaming-Performance (Vulkan/Mesa). Zudem konfigurieren wir Early KMS, um einen flackerfreien Übergang vom Bootvorgang direkt in deinen neuen Xfce-Desktop zu gewährleisten.

1. X-Server: Das grafische Fundament

Xorg bildet die Basis für die grafische Ausgabe. Wir installieren nur die notwendigen Kernkomponenten:

```
pacman -S xorg-server xorg-xinit
```

2. AMD-Grafiktreiber & Vulkan (Gaming-Setup)

Ein großer Vorteil von AMD-Grafikkarten unter Linux ist die native Unterstützung durch den Kernel-Treiber `amdgpu`. Während Nvidia-Nutzer oft auf proprietäre Treiber angewiesen sind, sind AMD-Treiber bereits fester Bestandteil des Kernels. Das sorgt für maximale Stabilität – auch nach Kernel-Updates.

Hinweis: Damit die `lib32`-Pakete installiert werden können, muss die `[multilib]`-Sektion in der `/etc/pacman.conf` aktiv sein, wie in Kapitel o. beschrieben.

AMD-Grafiktreiber & Vulkan
Basis-Treiber & OpenGL (Mesa):
<code>pacman -S mesa lib32-mesa xf86-video-amdgpu</code>
Vulkan für Gaming (Essentiell für Steam & Proton)
<code>pacman -S vulkan-radeon lib32-vulkan-radeon vulkan-icd-loader lib32-vulkan-icd-loader vulkan-mesa-layers lib32-vulkan-mesa-layers</code>
Video-Beschleunigung (Hardware-Decoding für YouTube/Filme):
<code>pacman -S libva-mesa-driver lib32-libva-mesa-driver</code>

3. Xfce Desktop & Login-Manager

Xfce gehört zu den stabilsten und ressourcensparendsten Oberflächen. Als „Türsteher“ für den grafischen Login nutzen wir LightDM:

```
pacman -S xfce4 xfce4-goodies materia-gtk-theme thunar-archive-plugin
```

```
pacman -S lightdm lightdm-gtk-greeter lightdm-gtk-greeter-settings
```

4. Early KMS | mkinitcpio

Durch **Early KMS** (Kernel Mode Setting) wird der Grafiktreiber bereits beim Booten in die RAM-Disk geladen. Dies verhindert **Black-Screens** und sorgt für einen flüssigen Übergang zum Desktop ohne Auflösungswechsel oder Flackern.

Die Konfiguration anpassen

Öffne die Datei `/etc/mkinitcpio.conf`:

```
nano /etc/mkinitcpio.conf
```

Suche die Zeile `MODULES=(...)` und trage den Treiber an erster Stelle ein:

```
MODULES=( amdgpu )
```

Boot-Image neu generieren

Wende die Änderung an, damit der Treiber beim nächsten Start sofort integriert ist:

```
mkinitcpio -P
```

... zur Übersicht



r. Desktop-Komfort: LightDM, Xfce und der Autologin

In diesem Schritt passen wir den Anmeldemechanismus an deine Bedürfnisse an. Wir konfigurieren den Display-Manager **LightDM**, um einen nahtlosen Übergang vom Bootvorgang direkt in deinen Xfce-Desktop zu ermöglichen.

Unser Ziel ist ein moderner, flüssiger Systemstart: Wir konfigurieren den **Autologin**, damit dich dein Rechner nach dem Einschalten ohne Umwege direkt an deinem Xfce-Arbeitsplatz begrüßt. Da aktuelle Hardware mit NVMe-Speicher oft schneller lädt, als die Grafikkarte initialisieren kann, bauen wir zudem eine intelligente Wartesekunde ein. So stellen wir sicher, dass der Monitor erst dann das Bild freigibt, wenn die Grafik-Treiber stabil bereitstehen – für einen sauberen Start ohne Flackern oder Fehlermeldungen.

1. Gruppe für den Autologin vorbereiten

LightDM erlaubt einen automatischen Login aus Sicherheitsgründen nur, wenn der Benutzer Mitglied der speziellen Gruppe **autologin** ist. In unserem Beispiel heißt der Benutzer **cat**. Wir erstellen die Gruppe und fügen ihn mit folgendem Befehl hinzu:

```
groupadd -r autologin
gpasswd -a cat autologin
```

Hinweis: Ersetze „cat“ durch deinen eigenen Benutzernamen, falls du einen anderen gewählt hast.

2. Lightdm konfigurieren

Wir passen nun die zentrale Konfigurationsdatei an. Öffne sie mit dem Editor:

```
nano /etc/lightdm/lightdm.conf
```

Suche die entsprechenden Abschnitte und nimm folgende Änderungen vor (entferne das # vor den Zeilen und trage deine Werte ein):

Abschnitt [LightDM]:

Sorgt dafür, dass das System wartet, bis die Grafikkarte bereit ist:

```
logind-check-graphical=true
```

Abschnitt [Seat:*]:

Hier hinterlegen wir deinen Benutzernamen und die gewünschte Sitzung (Xfce):

```
autologin-user=cat
autologin-user-timeout=0
autologin-session=xfce
```

3. Den grafischen Dienst aktivieren

Damit die Oberfläche beim nächsten Systemstart tatsächlich geladen wird, aktivieren wir den Hintergrunddienst:

```
systemctl enable lightdm
```

Abschließende Anmerkung:

Der Autologin erspart dir zwar das Passwort beim Hochfahren, schützt dein System aber weiterhin: Bei administrativen Aufgaben (z. B. Programminstallationen via sudo oder Pamac) wird dein Passwort nach wie vor abgefragt. Wer auf den Autologin verzichtet, landet stattdessen im klassischen Anmeldefenster und gibt dort sein Passwort ein.

[... zur Übersicht](#)



s. Dienste und Hintergrundprozesse aktivieren

Ein frisch installiertes Arch Linux ist zunächst ein "stilles" System. Damit Hardware-Komponenten wie dein Netzwerk-Adapter, das Bluetooth-Modul oder der Drucker nach dem Neustart sofort einsatzbereit sind, müssen die entsprechenden Hintergrund-Dienste (**Daemons**) im System registriert werden.

Dank **systemd** können wir diese Dienste bereits jetzt in der arch-chroot-Umgebung "scharf schalten". Sie werden dadurch fest in den Bootvorgang integriert und sorgen dafür, dass dein System beim Erreichen des Desktops bereits vollständig mit dem Internet verbunden ist und alle Wartungsaufgaben (wie die Mirrorlist-Optimierung oder die SSD-Pflege) im Hintergrund erledigt werden können.

1. Die zentrale Aktivierung

Mit einem einzigen Befehl aktivieren wir die wichtigsten Dienste für den automatischen Start, wobei `lightdm` bereits im vorherigem Kapitel bereits gestartet ist:

```
systemctl enable lightdm NetworkManager acpid avahi-daemon cups systemd-timesyncd  
reflector.timer bluetooth
```

2. Erläuterungen zu den Komponenten

- **lightdm**: Startet die grafische Benutzeroberfläche (Xfce). [1]
- **NetworkManager**: Initialisiert deine Internetverbindung. **Wichtig**: Achte hier penibel auf die Großschreibung von **NetworkManager**! [2]
- **acpid**: Verwaltet die Energieeinstellungen deiner Hardware.
- **avahi-daemon**: Ermöglicht die automatische Geräteerkennung im lokalen Netzwerk.
- **cups**: Stellt das Drucksystem bereit.
- **systemd-timesyncd**: Synchronisiert deine Systemzeit automatisch mit Atomuhren im Internet. [3]
- **reflector.timer**: Hält deine Spiegelservers-Liste für zukünftige Updates aktuell. [4]

3. Hinweis zur SSD-Pflege (NVMe)

Der Dienst `fstrim.timer` wird hier **absichtlich nicht** aktiviert. Da wir in der **fstab** (Kapitel k) bereits die Option **discard=async** eingetragen haben, nutzt dein System die modernere, kontinuierliche Methode zur SSD-Optimierung. Ein zusätzlicher wöchentlicher Trim-Dienst ist daher nicht erforderlich.

... zur Übersicht



t. Der Zündschlüssel: UEFI-Direktboot (EFISTUB)

In diesem finalen Schritt richten wir den Startvorgang deines Systems ein. Anstatt auf einen herkömmlichen Bootloader wie GRUB zu setzen, nutzen wir die modernste und puristischste Methode für UEFI-Systeme: den **EFISTUB**. Damit wird der Linux-Kernel direkt vom Motherboard gebootet.

Das Ergebnis ist ein extrem schlanker Bootprozess, der die Startzeit auf das absolute Minimum reduziert. Da wir die EFI-Partition direkt nach /boot gemountet haben, liegen alle notwendigen Dateien bereits an der richtigen Stelle.

1. Sicherheit geht vor: Das Recovery-Script (fix-boot.sh)

Wir führen den **EFISTUB**-Befehl nicht nur einmalig aus, sondern speichern ihn gleichzeitig als Script in einer ausführbaren Datei namens **/root/fix-boot.sh**. Sollte dein Boot-Eintrag jemals aus dem NVRAM des Mainboards verschwinden (z. B. nach einem BIOS-Update oder einer Systemwiederherstellung), kannst du ihn über das Live-System mit einem einzigen Befehl reparieren.

Wie die Wiederherstellung des Bootloaders mittels eines Arch-Livesystems und der fix-boot.sh durchgeführt wird, zeigt das Kapitel:

 Der Notfallplan: Boot-Recovery | EFISTUB-Reparatur

2. Die Startvorbereitungen abschließen

Bevor wir den Zündschlüssel umdrehen, stellen wir sicher, dass alle notwendigen Komponenten an ihrem Platz sind. Dieser Schritt ist wie ein kurzer **Pre-Flight-Check**: Wir vergewissern uns, dass die Werkzeuge für den UEFI-Boot bereitstehen und das System-Abbild (Kernel) alle aktuellen Treiber geladen hat.

⚠ **Wichtig:** Du musst dich zwingend noch in der **arch-chroot** Umgebung befinden!

a. Werkzeuge & Microcode bereitstellen: Wir stellen sicher, dass der Boot-Manager und die Prozessor-Updates installiert sind:

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

b. Das Start-Abbild (Initramfs) frisch backen: Mit diesem Befehl wird das Kernel-Image neu generiert, damit es alle Konfigurationen (wie das Btrfs-Modul oder deine Grafik-Treiber) für den ersten Start enthält:

```
mkinitcpio -P
```

3. Das Script `fix-boot.sh` erstellen und den Boot-Eintrag schreiben

Tippe die folgenden Befehle nacheinander ab. Nutze die **Profi-Checkliste** unten, um Tippfehler bei den Sonderzeichen zu vermeiden. Achte penibel auf die **doppelten Backslashes** (`\ \`).

a. Den Schreibvorgang einleiten:

Tippe diese Zeile und drücke **ENTER**:

```
cat << 'EOF' > /root/fix-boot.sh
```

Die Konsole springt in die nächste Zeile. Oft erscheint am Anfang ein Symbol wie `>` – das signalisiert: „Ich warte auf weiteren Text für die Datei“.

b. Den Datei-Inhalt (Script) eingeben:

Tippe nun nacheinander diese Zeilen ein und bestätige jede mit **ENTER**:

```
#!/bin/bash
```

Anschließend folgt der lange **efibootmgr**-Befehl (bitte ohne Umbruch in eine einzige Zeile tippen):

Den langen Befehl ohne Umbruch in eine einzige Zeile tippen:

```
D_DEV="/dev/$(lsblk -no PKNAME /dev/nvme*n1p2 | head -n 1)"; R_ID=$(lsblk -dno PARTUUID /dev/nvme*n1p2 | head -n 1); efibootmgr --create --disk "$D_DEV" --part 1 --label "Arch Linux" --loader /vmlinuz-linux --unicode "root=PARTUUID=$R_ID rw rootwait amdgpu.modeset=1 fbcon=nodefer plymouth.force-animation quiet splash loglevel=3 rd.systemd.show_status=auto rd.udev.log_level=3 initrd=\\amd-ucode.img initrd=\\initramfs-linux.img"
```

Erläuterungen zu diesem Einzeiler findest Du weiter unten.

⚠ **Wichtiger Hinweis zur Befehlseingabe:**

Der obige `efibootmgr` Befehl ist eine einzige, zusammenhängende Zeile. Aufgrund der Lesbarkeit im Dokument wird er am Seitenrand automatisch umgebrochen. Du musst ihn jedoch **ohne Unterbrechung** als eine einzige lange Zeile in das Terminal tippen. Drücke erst ganz am Ende (nach den letzten Anführungszeichen `"`) die Enter-Taste. Ein vergessenes Leerzeichen oder ein Tippfehler bei der PARTUUID hingegen führt dazu, dass das System nicht startet.

Das Terminal springt daraufhin in eine neue Zeile (markiert durch ein `>`), um auf den Abschluss der Datei zu warten.“

c. Die Eingabe abschließen

Tippe zum Abschluss nur diese drei Buchstaben und drücke **ENTER**:

```
EOF
```

Nach diesem letzten Tastendruck kehrt die Konsole zum normalen Prompt (root@archiso...) zurück. Die Datei wurde erfolgreich unter /root/fix-boot.sh gespeichert.

Der EFISTUB-Einzeiler im Detail

Der Befehl setzt sich aus drei logischen Teilen zusammen, die durch Semikolons (👉) getrennt sind:

1. **D_DEV="/dev/\$(lsblk -no PKNAME /dev/nvme*n1p2 | head -n 1)"**

- **Was es tut:** Sucht die physische Festplatte (z. B. nvme1n1).
- **Warum:** Da sich die Nummerierung (0 oder 1) ändern kann, fragt dieser Teil den Kernel direkt: „Auf welcher Platte liegt die Partition p2?“. Das Ergebnis wird in der Variable D_DEV gespeichert.

2. **R_ID=\$(lsblk -dno PARTUUID /dev/nvme*n1p2 | head -n 1)**

- **Was es tut:** Liest die **PARTUUID** deiner Root-Partition aus.
- **Warum:** Das UEFI benötigt diesen eindeutigen „Fingerabdruck“ der Partition, um Arch Linux beim Start zweifelsfrei zu finden – egal, welchen Namen die Platte gerade trägt.

3. **efibootmgr --create ...**

Hier passiert die eigentliche Magie. Die wichtigsten Schalter sind:

- **--disk "\$D_DEV" --part 1:** Sagt dem BIOS, dass der Bootloader auf der vorhin gefundenen Platte in der ersten Partition liegt.
- **root=PARTUUID=\$R_ID:** Übergibt dem Kernel die exakte Adresse deiner Systemplatte.
- **rootwait:** Der „Lebensretter“ für schnelle Systeme. Er zwingt den Kernel zu warten, bis die NVMe-SSD bereit ist.
- **fbcon=nodefer:** Sorgt dafür, dass das MSI-Logo deines Mainboards flackerfrei an Plymouth übergeben wird.
- **amdgpu.modeset=1:** Aktiviert die Grafiktreiber frühzeitig (Early KMS) für ein flackerfreies MSI-Logo.
- **quiet splash ...:** Schaltet die Textwüste beim Start stumm (Silent Boot).
- **initrd=\amd-ucode.img ...:** Lädt zuerst die Prozessor-Updates und dann das eigentliche System-Abbild. (Die doppelten Backslashes sind nur für die Erstellung des Skripts nötig).

💡 Technischer Hinweis zum Backslash:

Der doppelte Backslash (\ \) ist lediglich ein technischer Kniff für die automatische Erstellung der Datei mittels cat. Die endgültige Befehlsausgabe in der fix-boot.sh wird mit **einem** Backslash \ gesetzt.

4. Script aktivieren und Boot-Eintrag registrieren

Damit das Script auch wirklich funktioniert, mußt du danach noch die Rechte vergeben und es ausführen:

```
chmod +x /root/fix-boot.sh
/root/fix-boot.sh
```

Erfolgskontrolle:



- Mit dem Befehl **efibootmgr -v** kannst du prüfen, ob "Arch Linux" nun an erster Stelle der Boot-Reihenfolge steht.
- Um zu checken, ob die **fix-boot.sh** auch korrekt angelegt worden ist, gib folgenden Befehl ein.

```
nano /root/fix-boot.sh
```

5. 💡 Profi-Checkliste: Fehlerfreie Eingabe in der Konsole

Zeichen	Name	Tastenkombination (DE)	Zweck in diesem Befehl
'	Single Quote	Shift + #	Verhindert, dass die Shell das \$ sofort auswertet.
<<	Here-Doc	Shift + < (2x)	Startet die Block-Eingabe für die Datei.
>	Redirect	Shift + >	Erstellt oder überschreibt die Datei <code>/root/fix-boot.sh</code> .
\$	Dollar	Shift + 4	Leitet <code>\$(lsblk ...)</code> ein, um die PARTUUID zu finden.
_	Unterstrich	Shift + -	Wichtig für <code>amdgpu.modeset</code> und <code>amd-ucode</code> .
\\	Backslash	Alt Gr + ß	Erzeugt den Pfadtrenner für das UEFI im Script.

6. Abschluß und Überleitung

- **Backup:** Jetzt mit RescueZilla das Image ziehen. Das Skript `/root/fix-boot.sh` ist nun Teil deines Backups.
- **Speicherort:** `/root/fix-boot.sh` ist vor unbefugtem Zugriff geschützt, bleibt aber dennoch im chroot eines Live-Systems erreichbar.
- **Optionales Face-Lifting:** Wenn du beim Booten statt Text ein schönes **OEM-Logo** sehen möchtest, folge dem nächsten (optionalen) Kapitel u. Ansonsten fahre direkt mit Kapitel v. fort:
 Der Neustart - Dein System geht live fort.
- **Weiterführende Literatur:** Tiefergehende Details findest du im:
 Arch Wiki zu EFISTUB

... zur Übersicht



u. Optional: Dein MSI- & Arch-Logo im Rampenlicht

Wer kennt es nicht? Nach dem Einschalten flimmern hunderte Zeilen weißer Text über den Bildschirm. Das hat zwar Charme, wirkt aber bei einem modernen High-End-System oft unruhig. Mit **Plymouth** verleihen wir deinem MSI-System ein professionelles „Look & Feel“, wie man es sonst nur von perfekt abgestimmten OEM-Geräten kennt.

Anstatt der Textwüste wird direkt nach dem Start das edle **MSI-Logo** deines Mainboards angezeigt, ergänzt durch einen dezenten Arch Linux Ladekreis. Durch die Aktivierung von **Early KMS** und speziellen Boot-Parametern erreichen wir einen flackerfreien Übergang vom BIOS-Post direkt bis zum grafischen Login-Manager.

Wichtige Hinweise vorab:

- **Kompatibilität:** Diese Anleitung ist für aktuelle MSI AM5-Mainboards (B650, X670, X870) optimiert.
- **Flexibilität:** Du kannst dieses Kapitel jetzt direkt durchführen oder später nachrüsten. Da wir in Kapitel t. bereits die Datei `/root/fix-boot.sh` angelegt haben, bleibt dein „Zündschlüssel“ für den Notfall immer griffbereit auf der SSD liegen.
- **Rechte:** Wenn du später aus dem laufenden System heraus nachrüstest, setze einfach ein **sudo** vor die Befehle.
- **Bevor du weitermachst:** Stelle sicher, dass dein Board im **BIOS** optimal eingestellt ist. Diese Einstellungen (UEFI-Mode, Secure Boot Disabled, Memory Context Restore) hatten wir bereits in **Kapitel c. "BIOS/UEFI Vorbereitung"** vorgenommen.

1. Vorbereitung im arch-chroot

Stelle sicher, dass du dich noch in der arch-chroot Umgebung befindest und die Basis-Werkzeuge installiert sind:

```
pacman -S efibootmgr amd-ucode
pacman -S --needed base-devel git
```

2. Plymouth installieren (AUR)

Da Plymouth im Arch User Repository (AUR) liegt, nutzen wir einen temporären **builduser**, um das Paket sicher zu bauen:

a. Build-User anlegen:

```
useradd -m builduser
echo "builduser ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
```

b. Paket bauen und installieren:

Schritt-für-Schritt: Paket bauen und installieren
1. Den Ordner als builduser klonen:
<code>sudo -u builduser git clone https://aur.archlinux.org/plymouth /tmp/plymouth</code>
2. In den Ordner wechseln:
<code>cd /tmp/plymouth</code>
3. Das Paket bauen und installieren:
<code>sudo -u builduser makepkg -si --noconfirm</code>
4. Aufräumen: Entferne den temporären User wieder, um das System sauber zu halten:
<code>userdel -r builduser</code>
<code>sed -i '/builduser/d' /etc/sudoers</code>

3. Early KMS & Plymouth konfigurieren

Damit das Logo flimmerfrei erscheint, muss der Grafiktreiber geladen werden, bevor das System die restlichen Dienste startet.

Schritt-für-Schritt: Early KMS & Plymouth konfigurieren
1. Initcpio öffnen:
<code>nano /etc/mkinitcpio.conf</code>
2. Module ergänzen: Suche die Zeile <code>MODULES=(...)</code> und trage den Treiber ein:
<code>MODULES=(amdgpu)</code>
3. Hooks anpassen: Suche die Zeile <code>HOOKS=(...)</code> und füge <code>plymouth</code> direkt nach <code>udev</code> ein:
<code>HOOKS=(base udev plymouth autodetect modconf kms keyboard keymap consolefont block filesystems fsck)</code>
Speichere (Strg+O) und schließe (Strg+X).
4. Theme setzen: Wir wählen das Theme <code>bgrt</code> , welches das MSI-Logo deines Boards nutzt:
<code>plymouth-set-default-theme -R bgrt</code>
5. Image generieren: Wende die Änderungen auf das Boot-Image an:
<code>mkinitcpio -P</code>

4. Das Finale: Den Silent Boot scharf schalten

Da wir bereits in Kapitel t. unser Reparatur-Skript `/root/fix-boot.sh` mit allen wichtigen Parametern (wie `quiet`, `splash` und `rootwait`) vorbereitet haben, müssen wir den Boot-Eintrag jetzt nur noch einmal final in das Mainboard schreiben.

Tippe dazu einfach folgenden Befehl:

```
/root/fix-boot.sh
```

Der fertige Befehl in der **fix-boot.sh** sollte dann so aussehen (eine einzige Zeile):

Inhalt der fix-boot.sh

```
#!/bin/bash
```

```
D_DEV="/dev/$(lsblk -no PKNAME /dev/nvme*n1p2 \ | head -n 1)"; R_ID=$(lsblk -dno PARTUUID /dev/nvme*n1p2 \ | head -n 1); efibootmgr --create --disk "$D_DEV" --part 1 --label "Arch Linux" --loader /vmlinuz-linux --unicode "root=PARTUUID=$R_ID rw rootwait amdgpu.modeset=1 fbcon=nodefer plymouth.force-animation quiet splash loglevel=3 rd.systemd.show_status=auto rd.udev.log_level=3 initrd=\amd-ucode.img initrd=\initramfs-linux.img"
```

💡 Technischer Hinweis zum Backslash:**

Vielleicht ist dir aufgefallen, dass hier nur noch ein einfacher Backslash (`\`) verwendet wird. Der doppelte Backslash (`\\`) in Kapitel t. war lediglich ein technischer Kniff für die automatische Erstellung der Datei mittels `cat`.

... zur Übersicht



v. Der Neustart - Dein System geht live

Alle Komponenten sind installiert und für deine Hardware optimiert. In diesem letzten Schritt beenden wir die Wartungsumgebung und übergeben die Kontrolle vollständig an dein neues System auf der NVMe-SSD.

Wir stellen durch ein sauberes Aushängen der Partitionen sicher, dass alle Daten konsistent geschrieben wurden. Dank **EFISTUB** und **Early KMS** wird dein Rechner flackerfrei direkt in die grafische Oberfläche booten.

1. Die Installationsumgebung verlassen

Zuerst beenden wir die virtuelle Umgebung ...

```
exit
```

... und hängen die Laufwerke sicher aus:

```
umount -R /mnt
```

⚠ **Hinweis zur Datensicherheit:** Das **umount -R /mnt** ist bei Btrfs besonders wichtig. Es erzwingt, dass alle verbleibenden Schreibvorgänge aus dem Zwischenspeicher (Cache) fest auf die SSD geschrieben werden, bevor das System abgeschaltet wird.

2. Der finale Zündvorgang

Herzlichen Glückwunsch! Die Installation auf der Konsole ist abgeschlossen. Wir verlassen nun die Installationsumgebung und lassen dein System nun zum ersten Mal "fliegen".

Der Rechner-Neustart erfolgt abschließend mit:

```
reboot
```

Achtung: Ziehe den USB-Stick ab, sobald der Bildschirm schwarz wird.

3. Der erste Check im neuen Heim

Dein Rechner bootet nun direkt in den Xfce Desktop. Sobald du dort angekommen bist, öffne ein Terminal und führe eine erste Aktualisierung durch:

```
sudo pacman -Syu
```

Hier musst du nun dein Benutzerpasswort eingeben. Dieser Aufruf gleicht die Datenbanken sofort mit deiner neuen, optimierten Mirrorlist ab.

Willkommen in deinem neuen und pfeilschnellen Arch Linux! Du hast nun ein System ohne klassischen Bootloader, mit optimiertem BTRFS und automatisierter Wartung – ein echtes Unikat für jeden Arch-Fan.

Wie geht es weiter? (Post-Installation)

Nach dem erfolgreichen Start kannst du dein System weiter perfektionieren. In meiner ergänzenden Anleitung zeige ich dir, wie du Tools wie **zRAM** für optimale RAM-Nutzung, den AUR-Helper **Pamac** und ein individuelles Design einrichtest:

Eine PDF Anleitung zur Post-Installation wird auf der Seite zur Verfügung gestellt:

 [Arch Linux | Post-Installation](#)

... zur Übersicht