ADMIN:DOCS

Arch Linux: Best Practices und Tipps zur Systempflege

Leitfaden für die dauerhafte Wartung von Arch Linux: Rolling-Release-Hygiene, Arch News, .pacnew-Merging mit pacdiff, Orphan-Bereinigung, systemd-Audits und BTRFS-Snapshots.

Kategorie: Arch Linux Serie **Veröffentlicht:** 23.11.2024 **Zuletzt geprüft:** 27.08.2026

Nachdem du Arch Linux erfolgreich installiert und mit einer Arbeitsumgebung ausgestattet hast, beginnt die entscheidende Phase: der langfristige, stabile Betrieb. Arch Linux ist ein Rolling-Release-System ohne starre Versionszyklen. Das bedeutet, dass Softwarekomponenten fortlaufend auf den neuesten Upstream-Stand aktualisiert werden – vom Linux-Kernel über Systembibliotheken bis hin zur Desktop-Umgebung.

Ein Rolling-Release-System bricht nicht durch Updates, sondern durch fehlende Wartungsdisziplin: Wer ein Arch-System blind aktualisiert, geänderte Upstream-Konfigurationen (`.pacnew`) ignoriert oder verwaiste Pakete ansammelt, riskiert Boot-Probleme und inkonsistente Softwarestände.

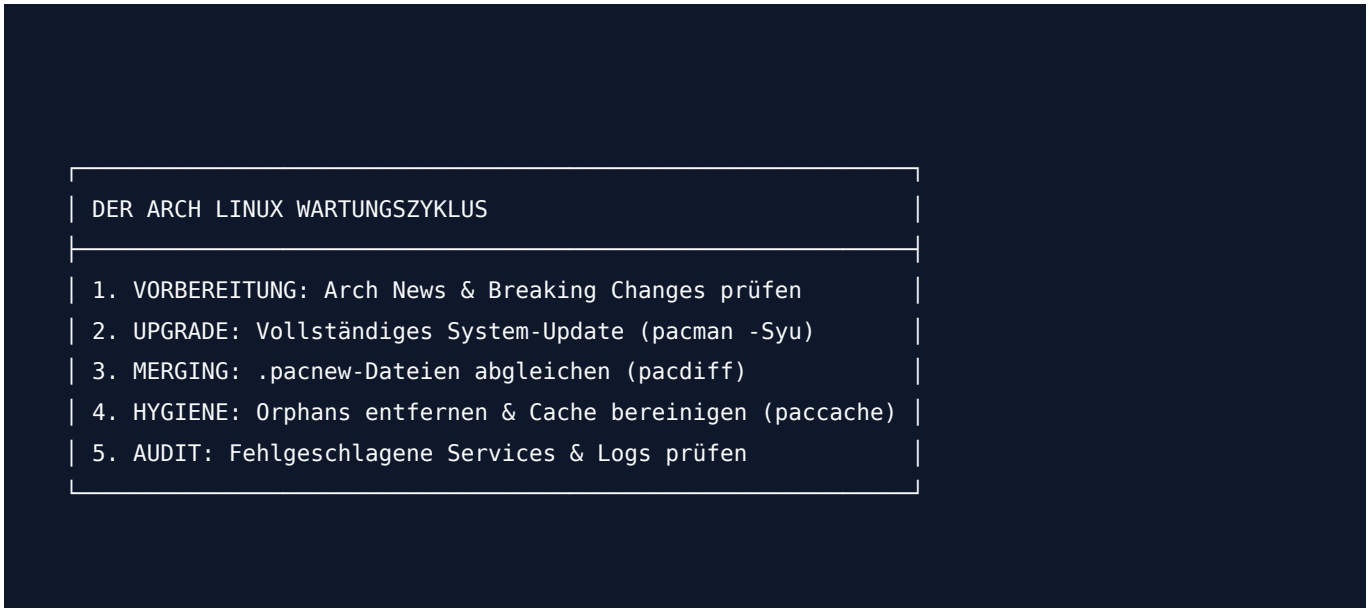
Dieser Leitfaden ist Teil unserer Arch-Linux-Serie und fasst die erprobten Best Practices für die professionelle Systempflege zusammen. Wir behandeln den vollständigen Wartungszyklus: von der Informationsbeschaffung vor Updates über das Konfigurations-Merging mit `pacdiff`, die automatisierte Cache- und Orphan-Bereinigung bis hin zu Snapshot-Rollback-Strategien mit BTRFS und `snapper`.

> **Voraussetzungen:** Ein lauffähiges Arch-Linux-System mit Sudo-Rechten. Grundlegendes Verständnis der Paketverwaltung mit Pacman sowie des Dienstemanagements über `systemd`.

Quelle: https://admindocs.de/de/arch-linux-serie/arch-linux-best-practices-tipps

Der Arch Linux Wartungszyklus

Systempflege unter Arch Linux folgt einer klaren, sequenziellen Routine:



1. Die professionelle Update-Routine

Ein Systemupdate unter Arch Linux besteht aus mehr als dem bloßen Ausführen eines Terminalbefehls. Um unvorhergesehene Inkompatibilitäten zu vermeiden, ist die Einhaltung standardisierter Schritte obligatorisch.

Arch News vor jedem Update konsultieren

Größere Arch-Upgrades (z. B. Migrationen des Kernels, von `glibc`, Python-Major-Releases oder Änderungen an Konfigurationspfaden) erfordern gelegentlich manuelle Eingriffe des Administrators (*Manual Intervention*). Diese Eingriffe werden von den Arch-Entwicklern vorab auf der offiziellen Website angekündigt.

Best Practices zur News-Prüfung:

1. **Webseite prüfen:** Vor dem Upgrade die Startseite von archlinux.org oder den RSS-Feed prüfen.
2. **Automatisierte CLI-Prüfung mit `informant`:** Das Tool `informant` aus dem AUR erzwingt das Lesen von unbestätigten Arch-News vor jedem `pacman`-Aufruf:

```
# informant installieren (via AUR-Helper oder makepkg)
yay -S informant

# News interaktiv im Terminal lesen und als gelesen markieren
informant read
```

Vollständige Systemupgrades mit `pacman -Syu`

Führe Upgrades ausnahmslos als vollständige Systemaktualisierung durch:

```
sudo pacman -Syu
```

> ⚠ **Partielle Upgrades sind strikt verboten:** Der Befehl `pacman -Sy <paket>` lädt die neueste Paketdatenbank herunter und installiert ein einzelnes Paket mit seinen aktuellen Abhängigkeiten, während der Rest des Systems auf altem Stand verbleibt. Dies führt unweigerlich zu inkompatiblen Shared Libraries (`.so` -Dateien) und zerstört die Systemintegrität.

Schnelle und konsistente Mirrors mit `reflector`

Veraltete oder desynchronisierte Paket-Spiegelserver (*Mirrors*) liefern inkonsistente Paketstände. Generiere wöchentlich eine frische Mirrorliste über HTTPS:

```
# reflector installieren
sudo pacman -S reflector

# Top 15 der schnellsten und aktuellsten deutschen HTTPS-Mirrors ermitteln
sudo reflector \
  --country Germany \
  --latest 15 \
  --protocol https \
  --sort rate \
  --save /etc/pacman.d/mirrorlist
```

Aktiviere den mitgelieferten systemd-Timer für automatisierte wöchentliche Updates der Mirrorliste:

```
sudo systemctl enable --now reflector.timer
```

LTS-Kernel als Rettungsanker (*Fallback*)

Ein Kernel-Update kann in seltenen Fällen Hardware-Regressionen verursachen (z. B. WLAN-Treiber, proprietäre Grafiktreiber). Installiere stets den Long-Term-Support-Kernel (`linux-lts`) als Ausweichoption parallel zum Mainline-Kernel:

```
# LTS-Kernel und Header installieren
sudo pacman -S linux-lts linux-lts-headers
```

Aktualisiere die Bootloader-Konfiguration (Beispiel GRUB):

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

Im Boot-Menü steht nun unter den *Erweiterten Optionen* jederzeit ein stabiler Ausweich-Kernel zur Verfügung.

2. Konfigurationsdateien mergen mit `.pacnew` und `.pacsave`

Wenn ein Paketupdate eine Konfigurationsdatei aktualisieren möchte, die du zuvor lokal verändert hast, überschreibt Pacman deine Anpassungen niemals blind. Stattdessen legt Pacman eine Datei mit der Endung `.pacnew` im selben Verzeichnis ab.

> **Warum `.pacnew`-Dateien kritisch sind:** Wer `.pacnew`-Dateien über Monate ignoriert, betreibt sein System mit veralteten Konfigurationsstrukturen. Geänderte Syntaxen in `/etc/pam.d/system-auth`, `/etc/sudoers` oder `/etc/systemd/` führen dazu, dass Dienste nach Updates den Start verweigern oder Sicherheitsmechanismen unwirksam werden.

`.pacnew`-Dateien aufspüren und verwalten mit `pacdiff`

Das Paket `pacman-contrib` liefert das interaktive Wartungswerkzeug `pacdiff` mit:

```
# pacman-contrib installieren
sudo pacman -S pacman-contrib
```

Starte die Überprüfung aller ungemergten Konfigurationsdateien:

```
# pacdiff mit bevorzugtem Diff-Tool (z. B. vimdiff, nvim -d, meld oder kdiff3)
DIFFPROG=vimdiff sudo -E pacdiff
```

`pacdiff` durchsucht das System nach `.pacnew`- und `.pacsave`-Dateien und bietet pro Treffer interaktive Optionen an:

```
==> pacnew-Datei gefunden für /etc/pacman.conf
:: [V]iew (Differenz anzeigen), [M]erge (Zusammenführen), [S]kip (Überspringen), [R]emove
(Löschen), [O]verwrite (Überschreiben), [Q]uit:
```

Best Practice für das Zusammenführen

1. **[V]iew:** Prüfe genau, welche Zeilen Upstream geändert wurden.
2. **[M]erge:** Übertrage neue Upstream-Optionen in deine produktive Konfiguration, während du deine individuellen Anpassungen (z. B. eigene Sudo-Defaults oder Pacman-Optionen) beibehältst.
3. **[R]emove:** Lösche die `.pacnew`-Datei erst, wenn alle relevanten Änderungen in die Originaldatei überführt wurden.

3. Systemhygiene & Paketdatenbank bereinigen

Mit der Zeit sammeln sich auf jedem Linux-System verwaiste Abhängigkeiten, temporäre Build-Reste und alte Paketversionen an.

Verwaiste Pakete (*Orphans*) entfernen

Wenn du ein Programm deinstallierst, verbleiben dessen Abhängigkeiten oft als verwaiste Pakete (*Orphans*) auf der Festplatte.

Ermittle alle verwaisten Pakete:

```
pacman -Qtdq
```

Entferne alle verwaisten Pakete mitsamt ihren ungenutzten Abhängigkeiten und globalen Konfigurationsdateien:

```
# Nur ausführen, wenn pacman -Qtdq Pakete listet
sudo pacman -Rns $(pacman -Qtdq)
```

> **Praxistipp:** Definiere einen Alias in deiner `~/.bashrc` oder `~/.zshrc`: `alias pacclean='sudo pacman -Rns $(pacman -Qtdq)'` So kannst du nach größeren Software-Deinstallationen mit einem Befehl aufräumen.

Fremdpakete und AUR-Altlasten auditieren

Mit `pacman -Qm` listest du alle Pakete auf, die nicht aus den offiziellen Repositories stammen (z. B. AUR-Pakete oder manuell installierte `.pkg.tar.zst`-Dateien):

```
pacman -Qm
```

Prüfe diese Liste monatlich: Veraltete oder verwaiste AUR-Pakete, die Upstream nicht mehr gepflegt werden, sollten deinstalliert oder durch offizielle Repository-Alternativen ersetzt werden.

Paketintegrität überprüfen

Überprüfe, ob installierte Dateien auf dem Dateisystem versehentlich verändert, beschädigt oder gelöscht wurden:

```
# Überprüft Dateigrößen und Prüfsummen aller installierten Pakete
sudo pacman -Qkk
```

Automatisiertes Cache-Management mit `paccache`

Pacman behält standardmäßig alle heruntergeladenen Pakete in `/var/cache/pacman/pkg/`. Ohne Wartung wächst dieser Ordner auf Dutzende Gigabyte an.

Verwende `paccache` aus `pacman-contrib`:

```
# Behalte die letzten 2 Versionen installierter Pakete (für Notfall-Rollbacks)
sudo paccache -r -k 2

# Entferne alle Cache-Dateien von bereits deinstallierten Paketen vollständig
```

```
sudo paccache -ruk0
```

Aktiviere die wöchentliche Bereinigung als systemd-Timer:

```
sudo systemctl enable --now paccache.timer
```

4. systemd-Wartung, Log-Management & Performance

Ein stabiles Arch-System erfordert regelmäßige Kontrollen der Systemdienste und des Speicherverbrauchs von Logdateien.

Fehlgeschlagene Dienste aufspüren

Prüfe nach jedem Systemstart und nach größeren Updates, ob alle systemd-Dienste sauber laufen:

```
systemctl --failed
```

Sollte ein Dienst als `failed` markiert sein, analysiere die Ursache gezielt im Journal:

```
journalctl -u dienstname.service -b -e
```

Log-Größe begrenzen und bereinigen

Das `systemd-journald`-Subsystem speichert Systemlogs persistent in `/var/log/journal/`. Begrenze den Speicherverbrauch in `/etc/systemd/journald.conf`:

```
[Journal]
Storage=persistent
SystemMaxUse=1G
SystemKeepFree=2G
```

Manuelles Bereinigen alter Logs:

```
# Behalte nur Logs der letzten 14 Tage
sudo journalctl --vacuum-time=2weeks

# Oder: Begrenze den Journal-Speicher sofort auf maximal 500 MB
sudo journalctl --vacuum-size=500M
```

Boot-Performance analysieren

Überprüfe, welche Dienste den Systemstart verlangsamen:

```
# Gesamtzeit des Bootvorgangs anzeigen
systemd-analyze

# Dienste sortiert nach Startzeit auflisten
systemd-analyze blame
```

```
# Kritische Kette von Abhängigkeiten beim Boot visualisieren
systemd-analyze critical-chain
```

SSD-Wartung mit TRIM aktivieren

Auf Systemen mit SSDs oder NVMe-Laufwerken stellt der TRIM-Befehl sicher, dass freigegebene Speicherblöcke vom Controller effizient wiederverwendet werden:

```
# Wöchentlichen fstrim-Timer aktivieren
sudo systemctl enable --now fstrim.timer
```

5. Disaster Recovery & Rollback-Strategien mit BTRFS und Snapper

Wer Arch Linux auf einem BTRFS-Dateisystem betreibt, kann Paketupdates vollständig absichern: Vor und nach jeder `pacman`-Transaktion wird automatisch ein Dateisystem-Snapshot erstellt. Schlägt ein Update fehl, bootest du direkt in den vorherigen Snapshot zurück.

Snapper und automatisierte Pacman-Snapshots einrichten

Installiere die erforderlichen Werkzeuge:

```
sudo pacman -S snapper snap-pac grub-btrfs
```

Erstelle eine Snapper-Konfiguration für das Root-Dateisystem:

```
sudo snapper -c root create-config /
```

Das Paket `snap-pac` klinkt sich automatisch als Pacman-Pre- und Post-Hook ein. Bei jedem `pacman -Syu` wird automatisch ein nummerierter Snapshot erstellt:

```
# Alle vorhandenen Snapshots auflisten
sudo snapper -c root list
```

Beispielausgabe:

```
 # | Type   | Pre # | Date                     | User | Cleanup | Description       |
Userdata
---+--------+-------+--------------------------+------+---------+-------------------
+---------
 0 | single |       |                          | root |         | current
|
 1 | pre    |       | Fri 27 Aug 2026 20:00:00 | root | number  | pacman -Syu
|
 2 | post   | 1     | Fri 27 Aug 2026 20:02:15 | root | number  | pacman -Syu
|
```

Snapshot-Boot über GRUB (`grub-btrfs`)

`grub-btrfs` generiert automatisch Boot-Einträge für alle vorhandenen BTRFS-Snapshots im GRUB-Menü.

Aktiviere den systemd-Dienst, der das Menü bei neuen Snapshots automatisch aktualisiert:

```
sudo systemctl enable --now grub-btrfsd.service
```

Sollte ein Update zu einem unbrauchbaren System führen, wählst du beim Booten einfach den Snapshot vor der Transaktion aus, bootest in ein funktionierendes System und führst ein Rollback durch:

```
# Rollback auf Snapshot 1
sudo snapper --ambit classic rollback 1
sudo reboot
```

6. Fortgeschrittene Tipps für den Alltag

Kleine, gezielte Optimierungen steigern den Komfort und die Zuverlässigkeit im täglichen Betrieb.

Early Microcode & Early KMS aktivieren

Microcode-Updates für Intel- und AMD-Prozessoren beheben kritische CPU-Sicherheitslücken (z. B. Spectre/Meltdown) und sollten so früh wie möglich beim Boot geladen werden:

```
# Je nach Prozessor installieren:
sudo pacman -S intel-ucode   # Für Intel-CPUs
# oder:
sudo pacman -S amd-ucode     # Für AMD-CPUs
```

GRUB erkennt die Microcode-Images automatisch bei der Neuerstellung der Konfiguration:

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

Für eine flackerfreie Grafik-Initialisierung beim Boot aktiviere das entsprechende Kernelmodul in `/etc/mkinitcpio.conf` (Early KMS):

```
# Beispiel für Intel-Grafik:
MODULES=(i915)

# Beispiel für AMD-Grafik:
# MODULES=(amdgpu)
```

Initramfs neu generieren:

```
sudo mkinitcpio -P
```

`pkgfile`: Fehlende Befehle und Dateien finden

Kennst du den Namen eines Befehls, weißt aber nicht, in welchem Paket er enthalten ist? `pkgfile` durchsucht alle Repositories:

```
sudo pacman -S pkgfile
sudo pkgfile -u
```

Beispiel:

```
pkgfile dig
# Ausgabe: extra/bind
```

Best-Practice-Wartungsmatrix

Intervall	Aufgabe	Befehl / Werkzeug
Vor jedem Update	Arch News prüfen	archlinux.org / `informant`
Täglich / Wöchentlich	Vollständiges Systemupgrade	`sudo pacman -Syu`
Nach jedem Update	Konfigurationsdateien mergen	`DIFFPROG=vimdiff sudo -E pacdiff`
Wöchentlich	Fehlgeschlagene Dienste prüfen	`systemctl --failed`
Wöchentlich	Verwaiste Pakete entfernen	`sudo pacman -Rns $(pacman -Qtdq)`
Automatisch	Paket-Cache bereinigen	`paccache.timer`
Automatisch	Mirrorliste optimieren	`reflector.timer`
Automatisch	SSD-TRIM ausführen	`fstrim.timer`
Monatlich	Fremdpakete / AUR auditieren	`pacman -Qm`, `arch-audit`

Intervall	Aufgabe	Befehl / Werkzeug
Monatlich	Journal-Größe kontrollieren	`journalctl --vacuum-time=2weeks`

Weiterführende Ressourcen

Arch Linux: Systemhärtung und Sicherheit – Best Practices Arch Linux: Fortgeschrittene Sicherheitsfeatures und Wartung Pacman: Der umfassende Leitfaden Arch Linux Installation und Grundkonfiguration Offizielle Arch Linux Wartungsrichtlinien (System Maintenance Wiki)

Fazit

Ein Rolling-Release-System wie Arch Linux verlangt keine stundenlange Pflege, sondern eine konsequente, strukturierte Routine. Indem du Systemd-Timer für Hintergrundaufgaben wie Cache-Bereinigung, Mirror-Updates und SSD-TRIM nutzt, reduzierst du den manuellen Aufwand auf ein Minimum.

Das Zusammenspiel aus automatischen BTRFS-Snapshots vor jedem `pacman`-Aufruf (`snap-pac`) und der direkten Boot-Integration (`grub-btrfs`) nimmt Upgrades das Risiko: Selbst wenn ein fehlerhaftes Paket den Start behindert, kannst du in Sekundenschnelle auf einen konsistenten Vor-Update-Zustand zurückrollen. Zusammen mit der regelmäßigen Bearbeitung von `.pacnew`-Dateien mittels `pacdiff` bleibt deine Konfiguration stets sauber und aktuell.

> **Praxis-Tipp:** Ignoriere `.pacnew`-Dateien niemals über längere Zeit. Wenn neue Syntax-Standards in Upstream-Diensten eingeführt werden, können veraltete Konfigurationen zu unerwarteten Dienst-Abbrüchen führen. Ein kurzer `pacdiff`-Check nach jedem größeren Update gehört zur Pflicht.

Im nächsten Artikel unserer Serie heben wir die Sicherheit deines Systems auf das nächste Level: In Arch Linux: Systemhärtung und Sicherheit – Best Practices härten wir Sudo und den PAM-Stack, sperren unberechtigte Prozess-Einsichten mit `hidepid`, konfigurieren eine restriktive Stateful Firewall mit `nftables` und sichern den SSH-Dienst gegen Brute-Force-Angriffe ab.