ADMIN:DOCS

Arch Linux: Der umfassende Leitfaden zum Paketmanager Pacman

Vollständiger Praxis-Guide zu Pacman: Repository-Konfiguration, Sync, Installation, sauberes Entfernen, Abhängigkeitsbäume, ALPM-Hooks und Troubleshooting.

Kategorie: Arch Linux Serie **Veröffentlicht:** 24.10.2024 **Zuletzt geprüft:** 27.08.2026

`pacman` (Package Manager) ist das zentrale Werkzeug und Herzstück jeder Arch-Linux-Installation. Entwickelt von Judd Vinet, kombiniert Pacman ein einfaches Binärpaketformat mit einem deklarativen, schnellen Abhängigkeitsmanagement über die Bibliothek `libalpm` (Arch Linux Package Management).

Pacman unterscheidet sich grundlegend von dpkg/apt oder rpm/dnf: Anstatt komplexe Pre- und Post-Installationsskripte beliebig tief zu verschachteln, setzt Pacman auf vorkompilierte `tar.zst`-Archive, strikte Metadaten und deklarative ALPM-Hooks. Die Administration ist direkt, transparent und extrem performant.

Dieser Leitfaden ist Teil unserer Arch-Linux-Serie und führt dich durch sämtliche Facetten von Pacman: von der Konfiguration in `/etc/pacman.conf` über grundlegende Operationen (Sync, Search, Install, Remove) bis hin zu fortgeschrittenen Themen wie ALPM-Hooks, Abhängigkeitsbäumen mit `pactree`, GPG-Schlüsselbundverwaltung und gezieltem Troubleshooting bei Paketkonflikten.

> **Voraussetzungen:** Ein lauffähiges Arch-Linux-System mit Sudo-Berechtigung sowie grundlegende Vertrautheit mit dem Linux-Terminal.

Quelle: https://admindocs.de/de/arch-linux-serie/arch-linux-der-umfassende-leitfaden-vom-paketmanager-pacman

Die Pacman- und ALPM-Architektur

Pacman teilt seine Aufgaben in klar definierte Schichten auf:



1. Konfiguration in `/etc/pacman.conf`

Die globale Steuerung von Pacman erfolgt über die Datei `/etc/pacman.conf`. Hier werden globale Optionen, Farbausgaben, Download-Parameter und die aktiven Repositories definiert.

Öffne die Datei zur Einsicht oder Bearbeitung:

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

Empfohlene globale Optionen

```
[options]
# Verhindert das versehentliche Entfernen überlebenswichtiger Kernpakete
HoldPkg     = pacman glibc

# Prüft vor Transaktionen, ob genügend Speicherplatz auf Zielpartitionen frei ist
CheckSpace

# Aktiviert farbige Terminalausgaben und den Pacman-Fortschrittsbalken
Color
ILoveCandy

# Parallele Downloads aktivieren (massiver Geschwindigkeitsvorteil)
ParallelDownloads = 5

# Strikte Signaturprüfung für maximale Sicherheit
SigLevel           = Required DatabaseOptional
LocalFileSigLevel  = Optional
```

- `ParallelDownloads = 5`: Lädt bis zu 5 Pakete gleichzeitig herunter, was Systemupdates drastisch beschleunigt.
- `ILoveCandy`: Ein beliebtes visuelles Easter Egg, das den Standard-Fortschrittsbalken in eine animierte Pacman-Animation verwandelt.
- `CheckSpace`: Berechnet vorab den Plattenplatzbedarf aller Zielmounts und bricht Transaktionen bei Speichermangel sicher ab.

Offizielle Repositories

Arch Linux strukturiert seine Software in klar getrennte offizielle Repositories:

```
# Das Fundament: Kernel, Init-System, Basis-Tools, Compiler
[core]
Include = /etc/pacman.d/mirrorlist

# Alle weiteren offiziellen Anwendungen: Desktop-Umgebungen, Browser, Server-Dienste
[extra]
Include = /etc/pacman.d/mirrorlist

# 32-Bit-Bibliotheken für 64-Bit-Systeme (z. B. für Steam, Wine, 32-Bit-Treiber)
[multilib]
Include = /etc/pacman.d/mirrorlist
```

> **Hinweis zum `[community]`-Repository:** Bis Mai 2023 existierte ein separates `[community]`-Repository. Arch Linux hat die Repository-Struktur konsolidiert und alle Community-Pakete vollständig in `[extra]` integriert. Der Eintrag `[community]` wird in modernen Installationen nicht mehr benötigt.

Mirrorlist optimieren mit `reflector`

Die Datei `/etc/pacman.d/mirrorlist` steuert, von welchen Servern Pacman Pakete lädt. Eine geographisch nahe und aktuelle Mirrorliste verhindert langsame Downloads:

```
# reflector installieren
sudo pacman -S reflector

# Top 10 deutsche HTTPS-Mirrors generieren
sudo reflector --country Germany --latest 10 --protocol https --sort rate --save
/etc/pacman.d/mirrorlist
```

2. Grundlegende Operationen: Die Hauptmodi

Pacman steuert seine Hauptoperationen über Großbuchstaben-Flags:

- `-S` (*Sync*): Pakete aus Remote-Repositories suchen, herunterladen, installieren und aktualisieren.
- `-R` (*Remove*): Installierte Pakete und Abhängigkeiten entfernen.
- `-Q` (*Query*): Die lokale Paketdatenbank nach installierten Paketen und Dateien durchsuchen.
- `-F` (*Files*): Dateien in Remote-Repositories suchen (Dateizuordnung).
- `-D` (*Database*): Metadaten installierter Pakete modifizieren (z. B. Installationsgrund).

Paketdatenbanken synchronisieren und Upgrades durchführen

```
# Vollständiges Systemupdate (Sync + Refresh + Upgrade)
sudo pacman -Syu

# Falls Repositories gewechselt wurden oder der Cache desynchronisiert ist:
sudo pacman -Syyu
```

> ⚠ **Niemals `pacman -Sy` ohne `-u` nutzen:** Das isolierte Synchronisieren der Datenbank (`-Sy`) gefolgt von der Installation eines einzelnen Pakets (`-S <paket>`) führt zu einem *Partial Upgrade*. Da neuere Bibliotheken installiert werden, während das restliche System veraltet bleibt, brechen bestehende Binaries (*Shared Library Mismatch*).

Sichere Update-Prüfung ohne Root-Rechte: `checkupdates`

Um zu prüfen, ob Aktualisierungen bereitstehen, ohne die lokale Systemdatenbank zu synchronisieren, verwende `checkupdates` aus dem Paket `pacman-contrib`:

```
# Prüft remote auf verfügbare Updates (sicher für Skripte und Statusleisten)
checkupdates
```

3. Pakete suchen und analysieren

Remote-Repositories durchsuchen (`-s`)

```
# Nach Paketen im Namen oder der Beschreibung suchen
pacman -Ss nginx

# Detaillierte Remote-Paketinformationen vor der Installation anzeigen
pacman -Si nginx
```

Beispielausgabe von `pacman -Si nginx`:

```
Repository      : extra
Name            : nginx
Version         : 1.26.1-1
Beschreibung    : Lightweight HTTP server and IMAP/POP3 proxy server
Architektur     : x86_64
URL             : https://nginx.org
Lizenzen        : BSD-2-Clause
Gruppen         : Nichts
Stellt bereit   : http-server
Hängt ab von    : pcre2  zlib  openssl  geoip  mailcap
```

```
Optionale Abh.  : Nichts
Konflikt mit    : Nichts
Ersetzt         : Nichts
Größe des Downloads: 540,22 KiB
Installationsgröße: 1,68 MiB
Packer          : Arch Linux Package Maintainers
Erstellt am     : Di 23 Jul 2024 14:10:00 CEST
```

Lokale Datenbank abfragen (`-Q`)

```
# Alle manuell und explizit installierten Pakete auflisten
pacman -Qe

# Alle als Abhängigkeit installierten Pakete auflisten
pacman -Qd

# Detaillierte Informationen zu einem bereits installierten Paket anzeigen
pacman -Qi nginx

# Alle Dateien auflisten, die zu einem installierten Paket gehören
pacman -Ql nginx

# Herausfinden, welches Paket eine bestimmte Datei auf der Festplatte bereitstellt
pacman -Qo /usr/bin/htop
# Ausgabe: /usr/bin/htop wird von htop 3.3.0-1 bereitgestellt
```

Abhängigkeitsbäume visualisieren mit `pactree`

Um die exakte Abhängigkeitshierarchie eines Pakets zu analysieren:

```
# pacman-contrib liefert pactree
pactree nginx

# Umgekehrter Baum: Welche installierten Pakete hängen von openssl ab?
pactree -r openssl
```

4. Installation und gezieltes Entfernen

Pakete installieren

```
# Einzelnes oder mehrere Pakete installieren
sudo pacman -S htop neovim git

# Paket nur herunterladen, aber nicht entpacken/installieren
sudo pacman -Sw wireshark-qt

# Lokale oder manuell gebaute Paketdatei (*.pkg.tar.zst) installieren
sudo pacman -U /pfad/zu/paket-1.0-1-x86_64.pkg.tar.zst
```

Pakete sauber und rückstandsfrei entfernen

Beim Entfernen von Software ist es entscheidend, verwaiste Abhängigkeiten und ungenutzte Konfigurationsdateien nicht auf dem System zu hinterlassen:

```
# Standard-Entfernung (Hinterlässt Abhängigkeiten und Konfigurationen)
```

```
sudo pacman -R paketname

# BEST PRACTICE: Entfernt das Paket, dessen ungenutzte Abhängigkeiten (-s)
# und globale Konfigurationsdateien (-n)
sudo pacman -Rns paketname
```

- `-s` (*Recursive*): Entfernt alle Abhängigkeiten, die von keinem anderen installierten Paket mehr benötigt werden.
- `-n` (*NoSave*): Löscht auch Konfigurationsdateien, anstatt `.pacsave`-Dateien anzulegen.

Installationsgrund ändern: `--asdeps` und `--asexplicit`

Wenn du ein Paket manuell installiert hast, das eigentlich nur eine temporäre Abhängigkeit ist (oder umgekehrt), kannst du den Status nachträglich korrigieren:

```
# Paket als Abhängigkeit markieren (wird bei Bereinigung als Orphan erkannt)
sudo pacman -D --asdeps libxml2

# Paket als explizit vom Administrator gewollt markieren
sudo pacman -D --asexplicit neovim
```

5. Systemwartung: Orphans, Cache & ALPM-Hooks

Verwaiste Pakete (*Orphans*) bereinigen

Orphans sind Pakete, die als Abhängigkeit installiert wurden, deren übergeordnetes Hauptprogramm aber inzwischen deinstalliert wurde:

```
# Alle verwaisten Pakete auflisten
pacman -Qtdq

# Alle verwaisten Pakete restlos entfernen
sudo pacman -Rns $(pacman -Qtdq)
```

Paket-Cache automatisiert verwalten mit `paccache`

Pacman behält alle heruntergeladenen Pakete in `/var/cache/pacman/pkg/`. Um Speicherplatz zu sparen, aber im Notfall eine Version zurückrollen zu können (*Downgrade*):

```
# Behalte die letzten 2 Versionen jedes Pakets
sudo paccache -r -k 2

# Entferne alle Versionen deinstallierter Pakete
sudo paccache -ruk0

# systemd-Timer für wöchentliche Bereinigung aktivieren
sudo systemctl enable --now paccache.timer
```

ALPM Transaktions-Hooks verstehen und erstellen

Pacman führt bei Transaktionen (Install, Upgrade, Remove) sogenannte ALPM-Hooks aus. System-Hooks liegen unter `/usr/share/libalpm/hooks/`, eigene Administrator-Hooks unter `/etc/pacman.d/hooks/`.

Beispiel: Automatischer Sicherheits-Audit mit `arch-audit` nach jedem Upgrade in `/etc/pacman.d/hooks/90-arch-audit.hook`:

```
[Trigger]
Operation = Install
Operation = Upgrade
Operation = Remove
Type = Package
Target = *

[Action]
Description = Prüfe installierte Pakete auf bekannte CVE-Schwachstellen...
When = PostTransaction
Exec = /usr/bin/arch-audit
```

6. GPG-Schlüsselbund & Troubleshooting

Fehler: "Ungültige oder beschädigte Paket-Signatur"

Tritt ein Signaturfehler auf, ist meist der lokale Arch-Schlüsselbund veraltet oder desynchronisiert:

```
# 1. Schlüsselbund aktualisieren
sudo pacman -Sy archlinux-keyring

# 2. Falls Schlüsselbund korrupt ist: Neu initialisieren
sudo rm -rf /etc/pacman.d/gnupg
sudo pacman-key --init
sudo pacman-key --populate archlinux
sudo pacman-key --refresh-keys
```

Fehler: "Fehler beim Sperren der Datenbank (db.lck)"

Wenn Pacman während einer Transaktion abbricht (z. B. durch Stromausfall oder `Ctrl+C`), bleibt die Sperrdatei `/var/lib/pacman/db.lck` bestehen.

```
# Prüfe zuerst, ob wirklich kein anderer Pacman-Prozess mehr läuft
pgrep -l pacman

# Falls kein Prozess aktiv ist: Lockfile sicher entfernen
sudo rm /var/lib/pacman/db.lck
```

Fehler: "Dateikonflikt: /usr/lib/... existiert im Dateisystem"

Wenn eine Datei existiert, die laut Datenbank zu keinem Paket gehört (z. B. durch frühere manuelle `make install`-Aufrufe):

```
# 1. Prüfen, wem die Datei gehört
pacman -Qo /pfad/zur/datei

# 2. Falls die Datei verwaist ist, Backup erstellen und überschreiben
sudo pacman -Syu --overwrite "/pfad/zur/datei"
```

Befehlsreferenz (Cheatsheet)

Aufgabe	Befehl
System aktualisieren	`sudo pacman -Syu`

Aufgabe	Befehl
Paket suchen	`pacman -Ss <suchbegriff>`
Paketinfo (Remote)	`pacman -Si <paket>`
Paket installieren	`sudo pacman -S <paket>`
Paket rückstandsfrei löschen	`sudo pacman -Rns <paket>`
Lokales Paket installieren	`sudo pacman -U <datei.pkg.tar.zst>`
Dateibesitzer ermitteln	`pacman -Qo /pfad/zur/datei`
Paketdateien auflisten	`pacman -Ql <paket>`
Orphans entfernen	`sudo pacman -Rns $(pacman -Qtdq)`
Cache bereinigen	`sudo paccache -r -k 2`
Updates sicher prüfen	`checkupdates`

Weiterführende Ressourcen

Arch Linux Installation und Grundkonfiguration Arch Linux: Best Practices und Tipps zur Systempflege Arch Linux: Systemhärtung und Sicherheit – Best Practices Arch Linux: Fortgeschrittene Sicherheitsfeatures und Wartung Offizielles Arch Wiki: Pacman Dokumentation Arch Wiki: Pacman Tipps und Tricks

Fazit

Der Paketmanager `pacman` ist das zentrale Nervensystem von Arch Linux. Wer seine Mechanismen, Flags und Konfigurationsoptionen beherrscht, besitzt das wichtigste Werkzeug, um ein Rolling-Release-System dauerhaft stabil, schlank und performant zu betreiben.

In diesem ersten Teil unserer Serie haben wir gelernt, warum Teil-Upgrades (*Partial Upgrades*) vermieden werden müssen, wie Signaturschlüssel über `pacman-key` verwaltet werden und wie verwaiste Abhängigkeiten und Paket-Caches sauber bereinigt werden. Mit diesem Rüstzeug bist du in der Lage, Paketkonflikte souverän zu lösen und dein System in einem optimalen Zustand

Quelle: https://admindocs.de/de/arch-linux-serie/arch-linux-der-umfassende-leitfaden-vom-paketmanager-pacman

zu halten.

> **Praxis-Tipp:** Nutze vor jedem System-Upgrade `checkupdates`, um anstehende Aktualisierungen risikofrei zu prüfen, und halte immer mindestens zwei Vorversionen im Cache, um im Notfall per `pacman -U` ein schnelles Downgrade durchführen zu können.

Im nächsten Artikel unserer Serie nutzen wir dieses Wissen für den Ernstfall: In Arch Linux Installation und Grundkonfiguration bauen wir ein vollständiges System von Grund auf auf – inklusive UEFI, BTRFS-Subvolumes, Festplattenverschlüsselung und Basis-Netzwerkkonfiguration.