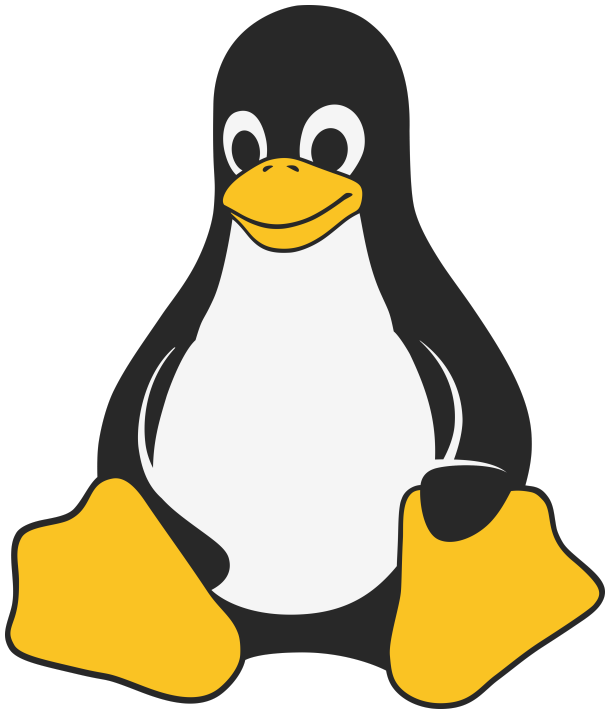
Unix/Linux-Systemarchitektur



Systemarchitektur von Unix und Linux

Die Linux-Systemarchitektur entspricht dem Unix-Architekturmodell. Die Konzepte sind dieselben. Unterschiede gibt es in der Implementierung und in den Details.

Viele der Konzepte und Prinzipien der Unix-Systemarchitektur stammen aus den 1970er Jahren. Sie finden sich auch in anderen modernen Betriebssystemen wieder, auch in Windows und macOS (die im Lauf der Jahrzehnte viele Unix-Konzepte übernommen haben).

Unix/Linux-Systemarchitektur in 3 Schichten

- User Space: Prozesse der Anwendungen und Dienste
- Kernel Space: Linux-Kernel
- Hardware/Geräte

Unix/Linux-Systemarchitektur in 4 Schichten

Oft wird die Systemarchitektur auch in 4 Schichten dargestellt. Die Shells müssen jedoch nicht als eigenständige Schicht betrachtet werden, sondern sie sind - technisch gesehen - selbst Anwendungen im User Space.

- User Space (Benutzer-Adressraum):
  - Prozesse der Anwendungsprogramme und Dienste
  - Prozesse der CLI-Shell und der grafischen Shell
- Kernel Space (Kern-Adressraum): Linux-Kernel
- Hardware/Geräte: Prozessor, Hauptspeicher, Massenspeicher, Netzwerk etc.

Anwendungen (Applications) im User Space

Die Anwendungen (CLI- und GUI-Anwendungen) sind die Schnittstelle des Benutzers zum System. Mit den Anwendungen interagiert der Benutzer und führt seine Aufgaben aus.

- CLI-Anwendungen (werden im Terminal aus der Shell (meist bash) gestartet): *ls*, *ps*, *grep*, *find*, *cat*, *nl*, *wc*, *sed*, *awk*, *vi* etc.
- GUI-Anwendungen (werden meist in der grafischen Oberfläche (Desktop) per Doppelklick gestartet): Dateimanager, Browser, Mail-Programm, Office-Programme, Bildbetrachter etc.

Dienste (Services) im User Space

Die Dienste (Services) sind Hintergrundprozesse ohne direkte Benutzerinteraktion, die bestimmte Aufgaben erfüllen, z.B.:

- Webserver (Apache, Nginx)
- Datenbankserver (MySQL, PostgreSQL)
- Dateiserver (Samba, NFS)
- SSH-Server
- Time-Server (NTP)
- etc.

Was ist ein Prozess?

Ein Prozess ist ein laufendes Anwendungsprogramm oder ein gestarteter Dienst. Ein Programm wird durch den Kernel in den Hauptspeicher geladen und ausgeführt. Ein Prozess hat einen eigenen virtuellen Adressraum, in dem er seine Daten und seinen Code ablegt. Ein Prozess hat einen Zustand (*starting*, *runnable*, *running*, *sleeping*, *waiting*, *zombie*) und Ressourcen (Speicher, Dateideskriptoren, CPU-Zeit etc.).

Prozesse (von Anwendungen und Diensten) im User Space

Die Prozesse laufen im User Space (Benutzeradressraum). Sie können nicht direkt auf die Hardware (Prozessor, Hauptspeicher, Massenspeicher, Netzwerk etc.) zugreifen, sondern sie beauftagen den Kernel, dies für sie zu tun. Der Kernel koordiniert die Hardwarezugriffe der Prozesse und stellt ihnen eine Aufruf-Schnittstelle (API) zur Verfügung (Systemaufrufe, System Calls).

Virtueller Adressraum

Jeder Prozess hat seinen eigenen virtuellen Adressraum, in dem er seine Daten und seinen Code ablegt. Der Adressraum eines Prozesses ist streng isoliert von den Adressräumen anderer Prozesse.

Ein Prozess kann nicht direkt auf den Adressraum eines anderen Prozesses zugreifen. Ein Prozess "sieht" nur seinen eigenen Adressraum. Die Adressräume anderer Prozesse "kennt" er nicht.

Aufgaben des Linux-Kernels (Kernel Space)

Der Linux-Kernel ist das Herzstück des Betriebssystems. Er ist zuständig für die Verwaltung der Hardware, die Koordination der Prozesse und für die Zuteilung von Ressourcen (Rechenzeit, Speicher, Dateideskriptoren etc) an die Prozesse.

Um diese Aufgaben zu erfüllen, benötigt er einen eigenen Adressraum, den Kernel-Space. Auf den Kernel-Space können die Prozesse nicht direkt zugreifen. Auch diesen Adressraum "sehen" die Prozesse nicht.

Aufgabenbereiche des Linux-Kernels

- Prozessverwaltung
- Speicherverwaltung
- Dateisystemverwaltung
- Interprozesskommunikation (IPC)
- Netzwerkverwaltung
- Geräteverwaltung
- Sicherheitsverwaltung

Auf die Aufgabenbereiche wird nachfolgend näher eingegangen.

Aufgabenbereich Prozessverwaltung

- Verwaltung von Prozessen (in einer Prozesstabelle)
- Zustände von Prozessen (starting, runnable, running, sleeping, waiting, zombie)
- Prozessressourcen (Speicher, Dateideskriptoren, Sockets, CPU-Zeit etc.)
- Zuweisung von CPU-Zeit (Scheduling, Time-Sharing)
- Prozessprioritäten
- Thread-Verwaltung (Ein Thread ist ein leichtgewichtiger Prozess, der innerhalb eines Prozesses läuft. Ein Prozess kann aus mehreren Threads bestehen.)

Aufgabenbereich Speicherverwaltung

- Zuweisung (und Freigabe) von Speicherressourcen (Code, Heap, Stack) an Prozesse.
- Memory Mapping: Abbildung des virtuellen Speichers der Prozesse auf den physischen Speicher.
- Swapping und Paging: Auslagerung von Speicherseiten auf die Festplatte, um Speicherplatz im Hauptspeicher freizugeben
- Memory Mapped Files: Abbildung von Dateien in den virtuellen Adressraum eines Prozesses

Aufgabenbereich Dateisystemverwaltung

- Abbildung von Dateien und Verzeichnissen auf Datenblöcke auf dem Massenspeicher (Festplatte, SSD)
- Dateisysteme (ext4, xfs, btrfs, zfs etc.)
- Zugriff auf Dateien und Verzeichnisse
- Rechteverwaltung

Aufgabenbereich Interprozesskommunikation (IPC)

- (Anonyme) Pipes: unidirektional, Reihenfolgegarantie, keine Repräsentation im Dateisystem, Kommunikation zwischen verwandten Prozessen
- Named Pipes (FIFOs): unidirektional, Reihenfolgegarantie, repräsentiert durch eine Datei im Dateisystem, Kommunikation zwischen beliebigen Prozessen
- Unix Domain Sockets: bidirektional, verbindungsorientiert, Reihenfolgegarantie, repräsentiert durch eine Datei im Dateisystem, Kommunikation zwischen beliebigen Prozessen

- Message Queues: asynchroner Nachrichtenaustausch, pub-sub (publish and subscribe)
- Shared Memory: gemeinsamer Speicherbereich für mehrere Prozesse, muss von den beteiligten Prozessen beim Kernel angefordert werden
- Semaphoren: Ampeln zur Synchronisation von Prozessen beim Zugriff auf gemeinsame Ressourcen

Aufgabenbereich Netzwerkverwaltung

- Netzwerkverbindungen mit TCP-Sockets: bidirektional, verbindungsorientiert, Reihenfolgegarantie
- Netzwerkverbindungen mit UDP-Sockets: bidirektional, verbindungslos, keine Reihenfolgegarantie
- Netzwerkprotokolle (IP, ICMP, TCP, UDP, ARP etc.)
- Netzwerkschnittstellen
- Routing
- Firewall

Aufgabenbereich Geräteverwaltung

- Treiberunterstützung (Character Devices, Block Devices)
- Erkennung und Konfiguration von Hardwarekomponenten
- Kommunikation mit externen Geräten

Aufgabenbereich Sicherheitsverwaltung

- Isolation der Speicherbereiche von Prozessen (so dass kein Prozess auf den Speicher eines anderen zugreifen kann)
- Benutzerverwaltung
- Rechteverwaltung
- Zugriffskontrolle (z.B. beim Zugriff auf Dateien: Ein Benutzer darf nur auf Dateien (lesend oder schreibend) zugreifen, für die er die Berechtigung hat.)
- Firewall

Links

- https://www.interviewbit.com/blog/linux-architecture/
- https://www.javatpoint.com/architecture-of-linux
- https://de.wikipedia.org/wiki/Linux_(Kernel)
- https://en.wikipedia.org/wiki/Linux_kernel
- https://en.wikipedia.org/wiki/Unix_philosophy