Kapitel 2

Kernel

Lernziele
Nach Abschluss dieses Kapitels,

1. kennen Sie die Aufgaben eines Kernels,
2. haben Sie eine Übersicht über Modularisierungs- und Schichtungsansätze,
3. kennen Sie die Grundstruktur des Linux-Kernels.

2.1 Modularisierung und Schichten

Große Software-Projekte werden meist in Module zerlegt. Dies gilt auch in der Regel für Betriebssystem. Wichtig ist nun die Frage, wie gegenseitige Modulaufrufe gestaltet werden können? Wie können dabei zirkuläre Beauftragungen vermieden werden?

Abb.: Modularer Aufbau ohne definierte Struktur

Bei komplexeren Architekturen bietet es sich an die Module in Abhängigkeit ihrer Aufgabe im Gesamtsystem in Schichten einzuteilen und dadurch die mögliche Kommunikation zwischen Modulen über die Regeln der Kommunikation zwischen Schichten und von Modulen in Schichten zu definieren.

Abb.: Modularer Aufbau mit definierten Schichten als Struktur

Für die Kommunikation bzw. Aufrufe der Module gelten folgende Regeln:

Diese Schichtung bringt Vorteile und auch einige Herausforderungen mit sich:

2.2 Kernel-Konzepte

Der Kernel als die erste Schicht über der Hardware muss nun eine Reihe von Modulen enthalten, die notwendig sind, um einen Rechnerbetrieb sicherzustellen.

Weitere (optionale) Module, die sich Aufgaben des Kernels kümmern, können sein:

Hierbei entsteht aber nun die architektonische Frage, wie diese Module an den Betriebssystemkern angebunden werden. Gemäß der Schichtenarchitektur werden sie in der gleichen Schicht laufen. Allerdings sind dann drei verschiedene Ansätze der Integration bzw. Kommunikation mit den oben aufgeführten Kernmodulen denkbar:

Der monolithische Ansatz ist heute der verbreiteste Ansatz für Betriebssysteme. Linux, Windows, BSD, MacOSX nutzen ihn. Microkernelsysteme sind (waren) Windows NT bis 3.51, AmigaOS, SymbianOS, QNX.

Der monolithische Ansatz für einen Kernel weist Vor- und Nachteile auf:

Obwohl die Vorteile des Microkernel-Ansatz theoretische überwiegen sollten, sind die häufigen Kontextwechsel zwischen User- und Kernel-Mode beim Microkernel-Ansatz eine beeinträchtigung für die Performance des Systems und damit ein ernsthafter Nachteil.

2.3 Der Linux-Kernel

Der Linux-Kernel geht auf die Arbeiten von Linus Torwald zurück, der einen unixoiden Betriebssystemkern für die X86 Architektur entwickeln wollte. In weiten Teilen ist der Kernel in C entwickelt worden. Nur Hardware spezfische Teile, die sich auf unterschiedliche Architekturen beziehen liegen in Assembler vor (Unterordner arch) der Source.

Die grundlegende Architektur des Kernels ist

In Bezug auf den Kernel muss zwischen zwei Schnittstellen unterschieden werden:

Der Linux-Kernel nutzt die Prozessorarchitektur wie folgt:

Die Sourcen von Linux sind frei zugänglich und damit kann auch selbst Linux oder Teile des Systems weiterentwickeln bzw. Beiträge zur Entwicklung leisten. Die Sourcen sind dabei wie folgt gegliedert:

Linux verwendelt Loadable Kernel Modules. Dadurch wird der Kernel flexibler und trotz der monolithischen Struktur müssen zum Systemstart nur die notwendigsten Komponenten geladen werden, da später Module nachgeladen werden kann.

Loadable Kernel Modules werden für folgende Bereiche eingesetzt:

Eine Übersicht über die Nutzung der Module im laufenden System kann man sich verschaffen mit: