Übungsblatt 01

André Reissig Benedikt Peterson

5. November 2025

Aufgabe 1

Wir geben erst eine kurze Erläuterung zu Makro- und Mikrokerneln. Danach werden wir die Vor- und Nachteile bzw. die Herausforderungen zu den einzelnen Teilen erklären.

Makrokernel Die Makrokernarchitektur ist besser bekannt unter dem Begriff Monolithical OS Kernel. Dabei hat man ein einzelnes Program im kernel mode. Das heißt, es gibt keine Separation von Betriebssystemkomponenten (Beispiele hierfür sind Scheduling, Filesystem, Networking, Device Treiber, Memory Management.). Sie ist die altbewährte Methode Betriebssysteme zu bauen.

Vorteil

- Höhere Performanz dadurch, dass alle Funktionen im selben Layer angesiedelt sind.
- Das input processing passiert auf dem interrupt level, was weniger latency zur Folge hat.
- Spin-System: IPC-Nutzung wird auf ein Minimum reduziert, um die Effizienz zu steigern (IPCs sind kostenintensiv.)

Nachteil

- Wenn eine Komponente abstürzt, stürzt das gesamte Betriebssystem ab.
- Betriebssystemcode wächst über die Zeit und wird schwer weiterzuentwickeln und zu warten.

Mikrokernel Hier hat der Kernel nur die notwendigen Funktionen : Die Verwaltung von Prozessen, Interprozesskommunikation (IPC) sowie grundlegende Schedulingfunktionen. Die Idee hinter Mikrokerneln ist, die Probleme der Makrokernel zu beheben. Also sie sollen flexibler werden durch die Auslagerung vieler Funktionen und gleichzeitig auch kleiner werden. Die Entwicklung der Mikrokernel ist noch nicht abgeschlossen und aktuell auch nicht praktikabel verglichen mit den Makrokernel. Während die erste Generation (Beispiel: External Pager) der Mikrokernel der Idee erlagen einen Makrokernel zu minimieren und dadurch häufig langsamer waren

als Makrokernel. Die zweite Generationen (Exokernel (lagert jedoch Device Driver nicht aus), L4) hat einen fruchtbareren Versuch gestartet, indem sie die Architektur des Mikrokernels komplett überdacht haben. Dort besteht die Hauptfunktion des Kernels in 3 Operationen: grant (Übergabe einer Page an einen anderen space), map (Zugang einer Page für einen anderen Space) und demap (entziehen des Zugangs abgesehen vom Owner der Page). Diese zweite Generation ist in der Realität noch nicht praktikabel aber zeigt großes Potenzial in der Art der Abstraktion.

Vorteil

- Alle anderen Funktionen sind nicht Teil des Kernels und können daher keinen Crash des Betriebssystems verursachen.
- Durch das Auslagern aller Funktionen auf eigene Server lassen sich Fehler besser finden.
- Ein kleinerer Kernel lässt sich einfacher verwalten und warten.
- Komponenten und Dienste können flexibler miteinander kombiniert werden.
- Betriebssystem kann einfacher von einer Hardware auf eine andere transportiert werden.
- Modularer System Struktur durchs Auslagern.

Nachteil

- Mikrokernel sind dokumentiert weniger performant zu sein. Der Grund ist die IPC. Laut [1, 2] ist dies jedoch nicht zwingend auf die Mikrokernelarchitektur zurückzuführen, sondern auf die Implementation der IPC. Trotzdem, bleiben Sie heutzutage tendenziell weniger preformant.
- Aktuell sind Mikrokernel noch nicht weit genug um praktikabel zu sein. Mehr Forschung ist von Nöten.

Zusammenfassend, haben Makrokernelarchitekturen das Problem, dass sie sowohl schwierig zu warten und zu erweitern sind, als auch das ein Fehler in einer Komponente schnell zum Absturz des Betriebssystems führen können.

Das versuchen Mikrokernelarchitekturen besser zu machen, indem Sie einige Prozesse in die user layer auslagern. Dadurch erzwingt man aber Kommunikation zwischen verschiedenen Layern, was mit einem Overhead einhergeht. Dieser lässt sich laut [1,2] mit Hilfe einer verbesserten Implementation beheben, aber scheinbar nicht komplett kompensieren. Das führt dazu, dass die viele gängigen Betriebssysteme wie Linux, FreeBSD in die Kategorie der Makrokernarchitekturen fallen. Darüber hinaus gibt es Hybridkernelsysteme, welche versuchen einen Kompromiss aus beiden zu bilden, um die Performance des Systems zu erhöhen. Dabei werden bestimmte Performacekritische Funktionen in den Kernel integriert, andere jedoch aus dem Kernel ausgelagert. Damit sind Hybridkernel zwischen den Monolithischen Kernels und den Mikrokerneln. Zu dieser Klasse gehören u.a. Windows und macOS.

Literatur

[1] Liedtke, Toward real µ-kernels, Communications of the ACM, 39(9):70–77,September 1996

[2] C. Maeda, B.N. Bershad, Networking performance for microkernels, Proceedings of Third Workshop on Workstation Operating Systems, 13:154 – 159, April 1992

[3] A.S. Tanenbaum: Modern Operating Systems, 2nd Ed. Prentice-Hall, 2001

[4] A. Silberschatz et al.: Operating Systems Concepts with Java, 6th Ed. Wiley, 2004

[5] B. Linnert, Vorlesungsfolien : Betriebssysteme WS 25/26

[6] https://de.wikipedia.org/wiki/Mikrokernel (as of Nov 04 2024)

Aufgabe 2

Die Programme sind zu finden unter dem folgenden link

`https://github.com/benedikt-code/bs_ws_25-26`

Die Dateien sind zu finden in dem Ordner `Uebung1`. Die Programme sind im Unterordner `src`.

Als verwendete Hilfsmittel ist ChatGPT zu nennen. ChatGPT wurde benutzt, um eine erste, grobe Idee zu bekommen, wie eine Implementierung aussehen könnte.