Betriebssysteme WiSe 2022/2023

Bearbeitung des 1. Aufgabenblattes 6. November 2022

Autoren: Elena Frank, Stefan Vesovic

1. (Architekturen)

   Ein Makrokernel ist ein Kernel, welches Funktionalitäten enthält, die eigentlich ausgelagert werden könnten. Beispiele für solche Funktionalitäten sind zum Beispiel der Netzwerkstack, Treiber oder Scheduler. Im Gegensatz dazu steht ein Mikrokernel. Dieser enthält nur Komponenten, welche nicht in externen Prozessen ausgelagert werden können. Zu essentiellen Funktionen gehört ein rudimentäres Speicherbereichsmanagment, Scheduling und ein IPC-Mechanismus.

   In einem Mikrokernel kann durch Isolation von komplexem Code im Userspace die Komplexität des Kernels niedrig gehalten werden. Darüber hinaus lässt sich durch das Auslagern in neustartbare Server die Sicherheit und Stabilität verbessern. Diese können im Fall eines Absturzes abgefangen und einfach neu gestartet werden. Dadurch bietet diese Art von Kernel auch mehr Flexibilität

   Traditionellerweise wurde behaupte, dass Mikrokernel wegen des mit dem IPC verbundenen Mehraufwand keine gute Performance ablegen können gegenüber Makrokernel. Jedoch fiel bei genauerer Untersuchung auf das trotz jahrelanger Optimierung nur ein geringer Teil, 3-7%, des Mehraufwandes durch Context switches entstand. Daraus ließ sich schließen, dass das Problem Architektur bedingt ist und durch eine neue bessere Architektur die Performance verbessert werden kann.

   Performance kann auch weiter optimiert werden durch bessere Nutzung von an der neuen Architektur angepassten Code. Da dort Code benutzt wurde, welcher für ein anderes Paradigma gedacht war, kam es zu großen Performance Einbußen. Zum Beispiel konnten die Performance des Netzwerkstacks von Mach deutlich verbessert werden, indem kleine Datenmengen direkt in die IPC-Pakete gepackt wurden, anstatt auf die Daten im Speicher des Servers zu verweisen oder Anfragen direkt über IPC zu machen und nicht über emulierte Syscalls.

2. (Steuerung von Geräten)

   Siehe hochgeladene Code-Dateien:

   - `print.c` / `print.h`: Datei welche `printf()` definiert, sowie Methoden um einzelne Zeichen von der seriellen Schnittstelle zu senden / empfangen.
   - `start.c`: Testprogramm, welches von der seriellen Schnittstelle liest und das empfangene Zeichen in einem formatierten String zurück echo-ed.
   - `Makefile`: Makefile zum kompilieren und linken.
   - `kernel.lds`: Linker Skript

   Ausführung:

   ```
   $ make
   $ qemu-bsprak -kernel kernel
   ```

   Github-Repo: https://github.com/elenaf9/WS2022-betriebssysteme