Einstieg in die Betriebssystementwicklung

- Grundlagen für das eigene OS -

Die Betriebssystementwicklung wird oft als die hohe Kunst der Informatik angesehen. Neben zahlreichen Betriebssystemen, die oft für sehr spezifische Aufgaben entwickelt worden sind, versuchen sich immer wieder der Hobbyinformatiker an der Entwicklung neuer Betriebssystemvarianten.

In diesem Workshop soll ein Einstieg in die Betriebssystementwicklung gegeben werden. Ziel ist es, einen kleinen Kernel zu entwickeln, der Lust auf mehr macht und Zuhause von den Teilnehmern beliebig erweitert werden kann.

Der Kernel wird für die x86 Plattform entwickelt. Hierfür existieren zum einen zahlreiche Emulatoren, die die Entwicklung erleichtern. Zum anderen kann der selbst entwickelte Kernel ohne weitere Hilfsmittel auf dem heimischen PC vorgeführt werden.

Für die Entwicklung wird der gcc und ein Editor nach Wahl eingesetzt. Der Einsatz von QEMU erleichtert uns die Arbeit. Ohne einen Emulator wie QEMU müssten wir bei jedem Entwicklungsschritt einen Testrechner neu starten.

Als Programmiersprache wird C eingesetzt. An einigen Stellen ist aber die Verwendung von Assemblerbefehlen unumgänglich.

Im ersten Schritt des Workshops wird auf den Bootvorgang sowie die Ausgabe von Text eingegangen. Um möglichst schnell 32-bit Code ausführen zu können, wird auf den GRUB-Bootloader zurückgegriffen. Am Ende dieser Phase soll ein erstes Hallo Linux Tage auf dem Bildschirm erscheinen.

Der zweite Teil geht auf die Anforderungen der x86-Architektur ein. Hier werden wir uns u.a. mit Interrupts beschäftigen. Auch werden die ersten Gerätetreiber für den neuen Kernel entwickelt. Für die weiteren Entwicklungsschritte ist ein Timer sowie ein Interruptcontroller unabdingbar. Je nach Kenntnisstand der Teilnehmer und zur Verfügung stehender Zeit besteht auch noch die Möglichkeit eine serielle Schnittstelle zu programmieren.

Im dritten und letzten Teil wird ein Ausblick über weitere Bestandteile eines Betriebssystems gegeben. Es wird auf den Umgang mit Arbeitsspeicher (insb. Paging und den Aufbau eines Heaps), Multitasking und den User Mode eingegangen.

Der Workshop orientiert sich an “JamesM's kernel development tutorials”, welches online verfügbar ist. So können die Teilnehmer die im dritten Teil besprochenen Punkte Zuhause umsetzen und das eigene Betriebssystem durch ein erstes Filesystem erweitern.

Um ein schnelles Vorankommen zu sichern, wird den Teilnehmern für jeden Arbeitsschritt ein C-File bereitgestellt. Dieses beinhaltet vor allem Hilfsfunktionen sowie ein äußeres Gerüst für die zu implementierende Logik. Die Teilnehmer sollen anhand von besprochenen Beispielen die Hardware initialisieren und z.B. eine Ausgabe erzeugen oder ein Gerät ansprechen. Bei den gestellten Aufgaben liegt der Schwerpunkt auf dem Verständnis der Arbeitsschritte sowie bei der Herangehensweise.

Von den Teilnehmern werden fortgeschrittene Programmierkenntnisse erwartet. Der Umgang mit Pointern und define-Anweisungen in C dürfen keine Probleme bereiten. Auch wird vorausgesetzt, dass die Teilnehmer mit einem Makefile und dem gcc umgehen können.

Grundkenntnisse der x86 Assemblersprache und Architektur sind von Vorteil, können aber auch während des Workshops angeeignet werden. Auch sind Grundlagen der Rechnerarchitektur oder Mikrocomputerprogrammierung für das Verständnis von Vorteil.

Über den Dozenten: *Tobias Stumpf studierte in Furtwangen technische Informatik. Er arbeitete bei der SYSGO AG in Mainz, welche sich auf Betriebssysteme für Echtzeitsysteme spezialisiert hat. Sein Schwerpunkt lag im Bereich der Kernelentwicklung des hauseigenen Betriebssystems, PikeOS. Aktuell setzt er sein Studium an der TU Chemnitz im Master Studiengang Informatik fort.*

Für die praktischen Übungen stehen den Teilnehmer zusätzlich Fernando Espinosa Arroniz und Matthias Keller zur Seite. Beide studieren an der TU Chemnitz Informatik und sind nebenberuflich im Bereich der hardwarenahen Softwareentwicklung tätig.