Laborversuch 2: Kernel Module

Lernziele:

In diesem Laborversuch sollen Sie lernen, wie man für den Linux Kernel nachladbare Funktionalitäten programmiert.

Erfolgskontrolle:

Um den Versuch testiert zu bekommen, müssen Sie vor Versuchsbeginn die Vorbereitungsfragen in Moodle beantwortet haben und mindestens 50% der Punkte haben. Am Ende des Versuchstages müssen Sie das fertige Kernel-Module laden können und Fragen zur Programmierung und Funktionsweise beantworten können.

Einführung:

Kernel Module können Sie auf den Raspberry PIs des Labors programmieren und nachladen, sie können das aber auch in Ihrer VM ihres Laptops. Je nach Linux-Distribution sind hierfür unterschiedliche Software-Pakete notwendig. Die Beschreibung für Ubuntu liegt unter: https://wiki.ubuntu.com/Kernel/BuildYourOwnKernel, einen Kernel muss man nicht komplett bauen sondern nur `make oldconfig` und `make modules_prepare` ausführen.

Sie können das Modul in `linux_module.tar` herunterladen, mit `tar xf linux_module.tar` entpacken und mit `make all` bauen. Danach können Sie das Module laden.

Ihre Aufgabe: schreiben Sie ein Kernel-Modul

Bauen Sie ihr eigenes Kernel-Modul `mod_kmalloc.ko`. Hierzu müssen Sie ein eigenes C-File anbieten und den `Makefile` anpassen.

Ihr Modul soll den Parameter „loop_cnt“ nehmen, der angibt wie häufig Speicher der Größe „alloc_size“ (ein weiterer Parameter) allokiert werden soll. Diese Speicherallokation (und nur diese) messen Sie mittels des Time-stamp-Counters mittels der Instruktion „rdtsc“ (auf Intel x86 / AMD64) bzw. mit hrtimers (ARM) und geben das auf der Kernel-Console mittels `printk()` aus.

Die Zeitmessung darf **nicht** in der Module-Initialisierung stattfinden, u.a. weil das system-nahe Programm `insmod` während der Initialisierung blockiert. Messen Sie in einem separatem Tasklet.
Erst wenn dieses Tasklet die Messung beendet hat, kann beim Entfernen des Moduls der Speicher wieder freigegeben werden.