5 Übungsaufgabe #6: Entwicklung eines PCI-Treibers

Im Rahmen von Aufgabe 6 soll für den im Seminar vorgestellten PCI Logikanalysator ein Linux-Gerätetreiber entwickelt werden. Dieser soll über Einträge im `sysfs` konfiguriert werden können und die abgetasteten Daten über eine Gerätedatei in `/dev` bereitstellen.

5.1 Grundlagen

Im Rahmen des Seminars wurde die Funktionsweise des Logikanalysator beschrieben und die Benutzung der in den Speicher eingeblendeten Geräteregister erklärt. Auch wurde ein kurzer Überblick über den Austausch der abgetasteten Daten zwischen Gerät und Treiber mit Hilfe von DMA gegeben.
Die für dieses spezielle Gerät nötige Initialkonfiguration der PCI-Schnittstelle findet sich ebenfalls auf den Seminarfolien.
Als weitere Informationsquelle für diese Aufgabe soll das Buch Linux Device Drivers[1] von Jonathan Corbet, Alessandro Rubini und Greg Kroah-Hartman dienen. Hierin sind besonders die Kapitel **3** *(Char Drivers)*, **10** *(Interrupt Handling)* und **12** *(PCI Drivers)* für diese Aufgabe relevant.

5.2 Funktionsumfang

Für diese Aufgabe soll ein Treiber im Kernelspace und eine Anwendung im Userspace implementiert werden.

Kernel-Treiber Der Treiber soll im `sysfs` die Möglichkeit bieten, die Abtastrate und die Trigger zu konfigurieren sowie das Abtasten zu starten. Außerdem soll der aktuelle Betriebsmodus des Logikanalysator abrufbar sein.
Die abgetasteten Daten sollen über ein zeichenorientiertes Gerät (*character device*) unterhalb von `/dev` an ein Benutzerprogramm übergeben werden. Bei einem `read()`-Aufruf auf die Gerätedatei müssen die vom Logikanalysator per DMA befüllten Speicherseiten hierzu in den Adressraum des aufrufenden Programmes (Userspace-Adressraum) kopiert werden.

Anwendung im Userspace Die Anwendung soll die Konfiguration des Logikanalysators vornehmen (z.B. aus Kommandozeilenparametern) und die abgetasteten Daten aus dem Gerät verarbeiten. Hierbei soll die Lauflängenkodierung der Signale dekodiert und eine Ausgabe erzeugt werden, bei der jeder Taktzyklus in einer eigenen Zeile dargestellt wird. Außerdem soll es möglich sein, mehrere (benachbarte) der 16 Signale zu einem „Bus" zusammenzufassen, welcher als ASCII oder Hex ausgegeben wird, das Ausgabeformat ist in den Folien nochmals beschrieben.

5.3 Zu Beachten

Die durch das Gerät bereitgestellten Speicherbereiche müssen vor der Verwendung erst eingeblendet werden. Da das Gerät einen Interrupt zur Signalisierung verwendet, muss die Unterbrechungsbehandlung registriert werden, bevor die Unterbrechungen aktiviert werden.

5.4 Weitere Hinweise

Die Kodierrichtlinien des Linux-Kernels müssen eingehalten werden. Synchronisierung und Speicherverwaltung müssen den üblichen Anforderungen wie der Abwesenheit von Wettlaufbedingungen (race conditions) und der Vermeidung (oder ggf. Minimierung) von Speicherlecks genügen. Oft sind im Kernel bereits Datenstrukturen oder APIs vorhanden, die den Programmierer beim Erfüllen dieser Anforderungen unterstützen. Sucht und benutzt diese bitte auch! Alles was Teil des Standard-Kernels ist ist auf jeden Fall erlaubt.
Zur Vereinfachung und da wir nicht genug Karten haben um das zu Testen darf angenommen werden, dass pro Rechner nur höchstens eine Logikanalysatorkarte existiert. Eine Überzahl an Karten soll jedoch sauber behandelt werden.

[1] `http://lwn.net/Kernel/LDD3/`

Aufgaben:

→ Lesen der Dokumentation, um die für den Treiber notwendigen Teile des Linux-Kerns zu verstehen und benutzen zu können.

→ Schreiben eines Treibers, der den Logikanalysator konfigurieren kann und einen minimalen Interrupthandler zur Verfügung stellt

→ Erweitern des Treibers um DMA-Funktionalität und Übergabe der physikalischen Adressen an den Logikanalysator

→ Implementierung eines zeichenorientierten Gerätes für den Zugriff auf die abgetasteten Rohdaten

→ Implementierung einer Userspace-Anwendung zur Nachbearbeitung der Rohdaten

→ Wie lautet der geheime Text (Bit 9-16)?

5.5 Abgabe: am 2016-02-04