# Projektbericht zur Vorlesung „Gerätetreiber“

Peter Conrad, 121528

25. September 2020

## 1 Einführung

Bei zahlreichen Notebook-Modellen erfolgt die Ausgabe von Grafik durch Grafikkarten der Firma Intel. Diese sind über den PCI-Bus mit dem System verbunden und verfügen in der Regel über ein fest definiertes Register zur Speicherung der momentanen Helligkeit des Bildschirms. Das Register befindet sich an der Adresse `0xF4`[1] und ermöglicht eine Steuerung der Helligkeit unabhängig davon, ob dies durch den verwendeten Grafikkarten-Treiber selbst unterstützt wird. Sollte der unter Linux zur Verfügung stehende Treiber eine Änderung der Helligkeit auf anderen Wegen nicht ermöglichen[2], kann diese unter Verwendung des Programms `setpci` modifiziert werden, indem die gewünschte Helligkeit als Byte-Wert in das genannte Register geschrieben wird. Unter der (häufig korrekten) Annahme, dass sich die Grafikkarte an der Adresse `00:02.0` befindet, bewirkt so der folgende Befehl eine Einstellung der Helligkeit auf den Maximalwert von 255:

```
$ setpci -s 00:02.0 F4.B=ff
```

Zielstellung des hier beschriebenen Projekts ist es, diesen Mechanismus als Gerätetreiber zu ipmlementieren. Dieser soll als Kernel-Modul für Linux zur Verfügung stehen und im System als Treiber für die Grafikkarte registriert werden, wobei als Testsystem ein Netbook vom Modell N220 der Firma Samsung zur Anwendung kommt. Der aktuelle Helligkeitswert soll durch entsprechende Dateien unterhalb von `/sys` les- und steuerbar sein. Hierfür sollen zwei Dateien angelegt werden, welche eine Änderung der Helligkeit auf einer Skala von 0 bis 255 oder 0 bis 8 ermöglichen. Bei letzterer soll der Helligkeitswert durch die Kurve $f(n) = 2^n - 1$, $0 \leq n \leq 8$ bestimmt werden, was beim verwendeten Testsystem einer subjektiv gleichmäßigen Änderung der Helligkeit entpsricht und eine einfache Verwendung durch Skripte ermöglicht.

## 2 Details zur Implementierung

### 2.1 Implementierung des PCI-Treibers

PCI (Peripheral Component Interconnect) ist eine Busarchitektur, welche der Verbindung von Peripheriegeräten mit dem Prozessor eines Computers dient.

[1] U.a. auch nachzulesen im Quelltext des i915-Moduls mit der Bezeichnung LBPC unter drivers/gpu/drm/i915/i915_reg.h

[2] In diesem Fall verliert das Register möglicherweise seine Wirkung.

Sie wurde als Ersatz für die ältere ISA-Schnittstelle entworfen, wobei die Zielstellung eine durch einen schnelleren Takt höhere Leistung, eine möglichst hohe Plattformunabhängigkeit sowei ein einfacheres Hinzufügen und Entfernen von Komponenten sind. Die detaillierte Standardisierung ermöglicht die Implementierung von Gerätetreibern auf einer höheren Abstraktionsebene als dies bei ISA der Fall ist. PCI kommt heutzutage in nahezu allen modernen Computersystemen zur Anbindung von Kompomenten zur Anwendung.

Unter Linux befinden sich die zur Programmierung von PCI-Treibern notwendigen Definitionen in der Datei `linux/pci.h`. Damit ein Kernel-Modul zur Steuerung eines PCI-Geräts zur Verfügung steht, muss dieses dem Kernel zunächst eine Liste der durch dieses Modul unterstützten Geräte mitteilen. Zur eindeutigen Identifikation verfügt jedes Gerät über Attribute in dafür vorgesehenen Registern. Diese umfassen mindestens die registrierte Hersteller-ID, eine vom Hersteller selbst gewählte Geräte-ID, sowie die Geräteklasse und optional (sollte es sich beim Gerät um eine Schnittstelle zu weiteren Geräten handeln) Hersteller- und Geräte-ID eines Subsystems. Für die Registrierung eines Moduls als Gerätetreiber ist lediglich die Angabe der Hersteller- und Geräte-ID erforderlich.

Für das verwendete Testsystem lauten die Hersteller-ID `0x8086` (Hersteller Intel) und die Geräte-ID `0xA011`. Diese müssen in der Struktur `pci_device_id` abgelegt werden, welche zur Angabe mehrerer möglicher Geräte wiederum in einem Array hinterlegt werden kann. Das Anlegen dieser Struktur wird durch den Makro `PCI_DEVICE` vereinfacht. Die fertige Liste der unterstützten Geräte muss nun mithilfe des Makros `MODULE_DEVICE_TABLE` exportiert werden, um dem Programm `depmod` die Auflistung aller durch Module unterstützter PCI-Geräte zu ermöglichen.

Im Anschlus daran muss sich das Modul bei seiner Initialisierung noch als Treiber beim System registrieren. Hierfür wird die Struktur `pci_driver` benutzt, welche mindestens die Angabe eines Namens, die zuvor definierte Liste der unterstützten Geräte sowie Zeiger zu zwei Funktionen `probe` und `remove` enthalten muss. Die Funktion `probe` wird dabei vom Kernel aufgerufen, wenn ein zu diesem Modul passendes PCI-Gerät gefunden wurde. Sie erhält als Parameter das Gerät in Form einer Struktur `pci_dev`, welches durch Verwendung der Funktion `pci_enable_device` aktiviert werden muss. Anschließend steht dieses Gerät dem Modul zur Verfügung. Die Methode `remove` wird aufgerufen, wenn das Gerät aus dem System entfernt wurde und ist für eventuelle Aufräumarbeiten zuständig. Insbesondere sollte hier die Funktion `pci_dev_put` aufgerufen werden, um dem Kernel mitzuteilen, dass das Gerät durch das Modul nicht weiter verwendet wird.

Der Aufruf der Funktion `pci_register_driver` mit der angelegten Struktur `pci_driver` zur Initialisierung des Moduls vervollständigt die Registrierung, sodass das Modul nach dem Laden zur Steuerung des Geräts zur Verfügung steht. Das Lesen und Schreiben des Konfigurationsregisters kann nun durch Aufruf der Funktionen `pci_read_config_byte` und `pci_write_config_byte` erfolgen.

## 2.2 Implementierung des sysfs-Interface

Zum Anlegen der Einträge in `/sys` ist die Erstellung eines *Kernel-Objekts* (Kernel Object) erforderlich. Diese Struktur ist in `linux/kobject.h` definiert. Die

Verwendung von Kernel-Objekten ist sehr komplex, kann jedoch stark vereinfacht werden, wenn lediglich das Anlegen von sysfs-Einträgen gewünscht ist. In diesem Fall kann die Funktion `kobject_create_and_add` benutzt werden, welche ein Kernel-Objekt anlegt zum System hinzufügt. Sie erfordert lediglich die Angabe eines Namens und einen Zeiger auf ein anderes Kernel-Objekt, welchem das anzulegende Objekt untergeordnet sein soll. Dazu bietet sich beispielsweise das globale `kernel_kobj` an, welches den Ordner `/sys/kernel` repräsentiert. Die Angabe des Namens `n220-backlight` bewirkt somit, dass die Dateien unterhalb von `/sys/kernel/n220-backlight` angelegt werden.

An die Erzeugung des Kernel-Objekts anschließend müssen nun noch die Attribute definiert werden, welche als Dateien sichtbar gemacht werden. Dies erfolgt durch Verwendung der Struktur `kobj_attribute`, welche mithilfe des Makros `__ATTR` erzeugt werden kann. Sie beinhaltet den Namen und Zugriffsrechte der azubildenden Variable in einer separaten Struktur `attribute` sowie Zeiger zu zwei Funktionen `show` und `store`. Die Funktion `show` wird beim Lesen der abgebildeten Datei aufgerufen, während `store` beim Schreiben aufgerufen wird.

Zur einfacheren Handhabung ist es nun möglich, die in den zuvor angelegten `kobj_attribute`-Strukturen hinterlegten `attribute`-Einträge in einem Array zu hinterlegen, welches wiederum in einer Struktur `attribute_group` eingebettet werden kann. Der Aufruf der Funktion `sysfs_create_group` mit dieser Struktur bewirkt das Anlegen der die beiden Helligkeitswerte repsärentierenden Dateien innerhalb des zuvor angelegten Ordners `/sys/kernel/n220-backlight`.

# 3 Probleme und Einschränkungen

Während die Erzeugung und Restrierung des PCI-Treibers relativ unkompliziert war, lagen die wesentlichen Probleme bei der Implementierung vor allem im Anlegen des für die sysfs-Einträge erforderlichen Kernel-Objekts. Die Dokumentation für Kernel-Objekte ist recht umfassend und beinhaltet zahlreiche Parameter und Funktionen, deren Herkunft und Bedeutung im gesamten Kontext nicht immer klar ersichtlich ist. So erfordert die Definition von Kernel-Objekten unter Umständen eine Struktur `kobj_type`, deren Attribute wiederum zahlreiche weitere Strukturen und Methoden beinhalten. Weiterhin werden Kernel-Objekte selbst innerhalb von *KSets* organisiert, deren Benutzung und Verwendung nochmals ein eigenes Kapitel gewidmet ist.[1]

Das korrekte Initialisieren von Kernel-Objekten ist somit auch nach dem Lesen der Dokumentation nicht klar ersichtlich. Der entscheidende Hinweis befand sich in der Kernel-Dokumentation[2] welcher erklärt, dass für das einfache Anlegen von sysfs-Einträgen die Benutzung der Funktion `kobject_create_and_add` in Verbindung mit `sysfs_create_file` oder `sysfs_create_group` ausreichend ist. Erst dieser Hinweis ermöglichte das erfolgreiche Anlegen der gewünschten Dateien.

Zuletzt muss für die Verwendung des vorliegenden Moduls noch eine wichtige Einschränkung genannt werden. Da es sich bei diesem Modul um einen Treiber handelt, welcher die Grafikkarte für sich beansprucht, darf für seine Benutzung der eigentliche Treiber nicht bereits geladen sein. Dies kann beispielsweise durch ein Hinzufügen von `module_blacklist=i915` zur Kernel-Kommandozeile im Bootloader des Systems bewirkt werden. Leider hat dies eine deutlich vermin-

derte Grafikleistung zur Folge, sodass es sich beim hier vorliegenden Projekt eher um ein Experiment als ein in der Praxis tatsächlich nützliches Modul handelt. Nichtsdestotrotz ergab sich durch die Programmierung dieses Moduls jedoch ein umfassender Erkenntnisgewinn zum Umgang mit dem PCI-System sowie dem sysfs-Interface des Linux-Kernels.

# Literatur


[1] Greg Kroah-Hartman Jonathan Corbet Alessandro Rubini. *Linux Device Drivers, 3rd Edition*. Bd. 3. `https://www.oreilly.com/library/view/linux-device-drivers/0596005903/` [abgerufen am 24.09.2020]. O'Reilly Media, Inc., 2005. ISBN: 9780596005900.

[2] Greg Kroah-Hartman. *Everything you never wanted to know about kobjects, ksets, and ktypes*. `https://www.kernel.org/doc/html/latest/core-api/kobject.html`. [online, abgerufen am 24.09.2020]. 2007.