Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Praktikum Betriebssysteme – TI3
Wintersemester 2012 – Praktikumsaufgabenblatt 4

Ein Kernelmodul zur Zeichenersetzung

Entwurfsdokument

Asmatullah Noor

Maschood Ahmad

Aufgabenbeschreibung:
Es ist ein Linux Kernel Treiber zu schreiben, dieser soll zwei Geräte anbieten. Ein Gerät soll einen Text kodieren, das andere Gerät soll einen kodierten Text dekodieren. Das Treibermodul trägt den Namen „translate“ , die beiden Geräte tragen die Namen „translate0“ und „translate1“ bei ersteres für das Codieren und das zweite fürs Decodieren zuständig ist. Die Majornummer soll bezogen werden, die Minornummern sollen gesetzte werden wobei das Gerät „translate0“ die 0 bekommt und das Gerät „translate1“ die 1. Beide Geräte sollen bereitstehen wenn das „translate“ Treibermodul geladen worden ist. Der Zugriff auf beide Geräte erfolgt über „/dev/translate[Minornummer].

B**eschreibung:**

Der Einstiegspunkt „module_init()“ des Treibers ruft die Initialisierungsfunktion „translate_init“ auf. In dieser wird die Struktur „translate_dev“ Initialisiert, welche aus dem Skulltreiber bekannt war. Ist die Struktur soweit initialisiert so wir sie an die Funktion „translate_setup_cdev“ übergeben. Dort wird nun zwischen den beiden Devices translate0 und translate1 (Geräten) per Index unterschieden bzw. werden dort beide Devices als Characterdevices mit unterschiedlicher Minornummer initialisiert.

Bei Nutzung der Devices wird auf die Struktur „file_operations“ zugegriffen, wir haben allerdings nur einige nötige Tag-Funktionen ausimplementiert, dies sind die Tags „owner“, „open“, „release“, „write“ und „read“. Somit kann der Zugriff sowie das Lese und Schreiben auf beiden Devices beschrieben werden, was alle notwendigen Verhalten für unsere Devices sind.

translate_open() wird aufgerufen wenn Zugriff auf ein Device notwendig ist. Über einen If-else-Baum und Prüfung der Semaphoren des Devices wird der zugriff gewährt oder der Fehlerwert „EBUSY“ zurückgegeben.

translate_release() wird aufgerufen wenn ein Device nicht mehr verwendet wird, dabei wird dann lediglich der Semaphore fürs Schreiben oder Lesen inkrementiert was den Zugriff für einen anderen Prozess erlaubt, da der Monitor wieder betretbar wird.

translate_write() wird aufgerufen wenn auf das Device „translate0“ geschrieben werden soll, da dieses Device fürs Codieren zuständig ist, wird für den Text im Buffer bzw. jeden Buchstaben des Textes, die Funktion zum Codieren aufgerufen und so der Text codiert.

Translate_read() wird aufgerufen wenn vom Device „translate1“ gelesen werden soll, da diese Device fürs Decodieren zuständig ist, wird für den Text im Buffer bzw. jeden Buchstaben des Textes , die Funktion zum Decodieren aufgerufen und so der Text decodiert.

Eigene Funktionen haben wir drei, dabei handelt es sich um die Funktion zum Codieren, Dekodieren und um den Index eines Buchstabens zu ermitteln. Dabei werden beim Prüfen der Buchstaben aus dem Buffer Offsetwerte genutzt die hartcodiert im Headerfile stehen, daher muss der „subst“ passen, wie unter den Hinweisen schon beschrieben. Eine dynamische Anpassung des Offsets an den genutzten „subst“ findet nicht statt.

Der Ausstiegpunkt „module_exit“ wird beim Entladen des Moduls aufgerufen, dabei wird dann sämtlicher allokierter Speicher aus der Initialisierung beider Devices freigegeben, die Pointer NULL gesetzt, die angemeldeten Charakterdevices gelöscht und das Modul deregistriert.