Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Betriebssysteme Praktikum 4

Entwurf

Florian Heuer

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Beschreibung

In diesem Laborversuch soll ein Treiber für ein Character Device unter Linux entwickelt werden. Dieser Treiber soll über ein Kernelmodul in den Kernel geladen werden. Zudem soll der Treiber 2 Character Devices steuern. Die Aufgabe des Treibers ist es den aus der Aufgabenstellung zu entnehmenden Automaten mit implementieren. Über schreibenden Zugriff sollen die Befehle zum Zustandswechsel übergeben und über lesenden Zugriff der aktuelle Zählerstand erhalten werden. Des Weiteren soll ein Installationsscript erstellt werden, dass die Installation des Treibers vornimmt und die 2 Devices erzeugt.

2 Entwurf

2.1 Umgebung

Das Kernelmodul wird in der Sprache C entwickelt. Als Compiler kommt „GCC“ zum Einsatz. Das Betriebssystem Open SUSE in Version 12.3 wird zum kompilieren und ausführen des Codes benutzt. Als Editor wird Sublime 2 verwendet. Zudem wird zur Erleichterung des Kompiliervorgangs ein Makefile erstellt, welches das Kernel Modul erzeugt.

2.2 Konzept

2.2.1 Aufbau des Kernelmoduls

Das Kernelmodul besteht aus den Dateien timer.c und timer.h. Die Headerdatei definiert wichtige Konstanten und Strukturen unter anderem die Transitionen und Zustände des Automaten. Der Automat wird über eine Struktur abgebildet.
Die Sourcedatei sorgt durch die Methoden module_init und module_exit, dass die Device Nodes erstellt und mit einer entsprechen dynamisch erzeugten Majornumber versorgt werden. Die Unterscheidung der beidem Devices erfolgt über die Minornumber. Des Weiteren müssen noch die open, close , read & write Methoden implementiert werden um den Zugriff auf das Device zu steuern. Die Devices sollen über einen Treiber gesteuert werden

also muss sichergestellt werden, um welches Device es ich handelt. Das passiert in der „device open“ Funktion. Hier wird anhand der von MKDEV() vergebenen Nummer und dem der „open“ Funktion übergebenen inode Parameter abgeglichen um welches Device es sich handelt. Das ist relevant weil für die 2 Devices ein etwas unterschiedliches Verhalten implementiert werden soll.

2.2.2 Installationscript

Da das Modul die Erstellung der Devicenodes und die dynamische Erzeugung der Majornumber, sowie das Löschen der Devicenodes vornimmt, übernimmt das Installationsscript nur die Ein- bzw. Ausbindung des Kernelmoduls.

2.2.3 Aufbau des Automaten

Der Automat wird über eine Struktur abgebildet. Jedes Device erhält eine eigene Strukurvariable. In dieser sind der Name, die Devicenummer, der aktuelle Zustand und mehrere Zählervariablen, welche für die Timer Ausgabe benötigt werden, enthalten.

Die Zustandswechsel erfolgen über den schreibenden Zugriff auf das Characterdevice. Die übergebenen Befehle werden ausgewertet und lösen entsprechende Aktionen für den zu steuernden Timer aus. Die Ausgabe erfolgt über lesenden Zugriff auf das Characterdevice.

3 Repository

Der komplette Sourcecode des Programms “**kernelStopWatch**” ist unter folgendem Link auf GitHub zu finden.

https://github.com/FlowwX/kernelStopWatch