Betriebssysteme - Entwurfsdokument
Aufgabe 4
Gruppe: Kai Brusch, Matthias Nitsche

In der Aufgabe sollen wir ein Kernel-Modul für die Codierung von Zeichenketten programmieren. Dies ist relativ nahe an den vorgegebenen Scull Beispiel angelehnt. In der Header Datei sind alle relevanten Konstanten und Structs abgelegt für den lese und schreib Zugriff. In diesem Treiber nutzen wir translate0 um eine Zeichenkette zu codieren nach einem spezifischen vorgegebenen Alphabet und mit translate1 wieder zu dekodieren.

Am Start des Programms wird translate_open genutzt um die Semaphoren zu blockieren („lock“) und startet die Prozedur oder gibt zurück das dieser gerade blockiert ist. Das Translate_release ist dabei das Gegenstück, welches den Semaphor nach Beendung eines Zugriffs freigibt. Wenn es sich bei dem Device um translate0 handelt, wird die eingabe verschlüsselt. Die maximale Größe des Buffers für die Eingabe ist auf 40 Chars begrenzt. Wenn der Buffer Volläuft wird der Zugriff unterbrochen. Über translate1 wird der Buffer dann wieder dekodiert und in den user_space abgelegt. Dies ist nach Scull implementiert.

Um die Devices zu benutzen, bzw. file_operations anzuwenden, benutzen wir translate_open, welches Prüft ob auf das Device zugegriffen werden kann, andernfalls ist das Device als (BUSY) gekennzeichnet und ein Fehlerwert zurückgegeben.

Die eigentliche Umsetzung der codierung und decodierung erfolgt über einen Substr. Dieser Substring repräsentiert zwei Alphabete, hier einmal das Großbuchstaben und Kleinbuchstaben Alphabet. Es wäre aber möglich auch andere Alphabete zu nutzen. Die Codierung der Kleinbuchstaben ist bis len(substr)/2 und die Dekodierung für die Großbuchstaben dementsprechend von len(substr)/2 bis len(substring). Dies alles erfolgt über die ASCII Nummer der einzelnen Characters.

Das installieren und deinstallieren (install.sh und uninstall.sh) folgt wie in Ihren Beispielen (Scull) beschrieben, mit dem Unterschied das wir zusätzlich beim installieren noch das Makefile aufrufen.