4 Ein Kernelmodul zur Zeichenersetzung

Schreiben Sie einen Gerätetreiber für zwei Geräte mit den Device-Nodes `/dev/translate0` und `/dev/translate1`, die eine Zeichenersetzung gemäß einer Ersetzungstabelle vornehmen, die in dem globalen statischen String `subst` enthalten ist. Dieser String enthält die Ersetzungszeichen für die Buchstaben 'a' ...'z' gefolgt von den Ersetzungszeichen für 'A' ...'Z'. Der Vorgabewert dieses Strings lautet
`"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"`.
Damit werden also Klein- in Großbuchstaben sowie Groß- in Kleinbuchstaben gewandelt.

Das Device `/dev/translate0` soll als Device-Node mit der Minor-Nummer 0 angelegt werden, und die Zeichenumsetzung in Vorwärtsrichtung machen, während das Device `/dev/translate1` die Minor-Nummer 1 erhält und die Rückwärtsumsetzung durchführt.

Zur *Vorwärtsumsetzung* beim *Schreiben* eines Kleinbuchstaben auf `/dev/translate0` wird der Ausdruck `zeichen - 'a'` als Offset in den String `subst` genommen, und das dortige Zeichen als umcodierter Wert im Buffer gespeichert. Für Großbuchstaben ist der Offset sinngemäß anzupassen.

Beim *Lesen* von `/dev/translate0` wird der gespeicherte Bufferwert unverändert ausgegeben.

Beim *Schreiben* eines Buchstaben auf `/dev/translate1` wird der Buchstabe unverändert in den Buffer dieses Devices geschrieben. Zur *Rückwärtsumsetzung* beim *Lesen* eines Buchstaben von `/dev/translate1` wird das erste Vorkommen dieses Buchstaben im Substitutionsstring mit der Bibliotheksfunktion `strchr()` gesucht, und der Offset der Fundstelle in einen Buchstaben umgesetzt. Beispiel: Der Buchstabe 'C' wird im Vorgabe-Substitutionsstring beim Offset 2 gefunden. Diese Position gehört zum Buchstaben 'c'. Der so gewandelte Buchstabe wird in den Buffer oder einen Hilfsstring geschrieben und mit `copy_to_user()` an den lesenden Prozess übertragen.

Es sollen nur die „normalen" ASCII-Zeichen, also a–z und A–Z codiert werden, alle anderen, also insbesondere Satzzeichen, Ziffern und nationale Sonderzeichen wie ä, ö, ü und ß sollen unverändert übernommen werden.

- Zur *Vorwärtsumsetzung* schreibt man also Klartext auf `/dev/translate0`, und erhält beim Lesen von diesem Device den umcodierten Text. Zur *Rückwärtsumsetzung* schreibt man den umcodierten Text auf `/dev/translate1`, und erhält beim nächsten Lesen den Klartext.
- Die beiden Geräte sollen von einem gemeinsamen Treibermodul gesteuert werden, die Funktion wird durch die *minor device number* festgelegt:
- Minor number 0 bedeutet *Vorwärtsumsetzung*
  Minor number 1 bedeutet *Rückwärtsumsetzung*.
- Die *major device number* soll dynamisch vom Kernel vergeben werden.
- Der Treiber muss beim Laden des Moduls einen Pufferspeicher allozieren, der beim Entfernen des Moduls wieder freigegeben wird.
- Es darf jeweils nur ein Prozess das Gerät zum Lesen und zum Schreiben öffnen. Weitere Versuche, das Gerät zu öffnen, werden mit `-EBUSY` abgewiesen.

- Der Leseprozess blockiert, wenn der Puffer leer ist, der Schreibprozess blockiert, wenn der Puffer voll ist (Tipp: Spendieren Sie ein Strukturmember `fillcount`, das die Anzahl der Zeichen im Buffer zählt). Die Lese- und Schreibzeiger werden zyklisch durch den Puffer bewegt.
- Die Buffergröße `translate_bufsize` und der Substitutionsstring `translate_subst` sollen als Modulparameter festgelegt werden können. Beim Laden ohne Parameter sollen Standardwerte genommen werden: `translate_bufsize = 40` und `translate_subst =`
  `"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"`.
  (Wie Ihnen sicherlich schon klar war, werden damit werden Groß- und Kleinbuchstaben vertauscht.)
- Alles was Sie zur Bearbeitung der Aufgabe wissen müssen steht in den Kapiteln 1 – 5 sowie im Kapitel 6, Abschnitt „Blocking I/O“ des Buches „Linux Device Drivers“ von Jonathan Corbet, Alessandro Rubini, und Greg Kroah-Hartman, die URL des ersten Kapitels ist `http://oreilly.com/catalog/linuxdrive3/book/ch01.pdf`.
- Die zugehörigen Beispielprogramme finden Sie unter der URL `http://examples.oreilly.com/linuxdrive3/examples.tar.gz`. Ich habe die Programme aus dem Ordner `scull` so modifiziert, dass sie sich auf unseren virtuellen Maschinen kompilieren lassen und auf meinem Pub-Verzeichnis abgelegt: `https://pub.informatik.haw-hamburg.de/home/pub/prof/fohl/Bs/Praktikum/ldd_example_scull.tar.bz2` Schauen Sie sich insbesondere das Programm `pipe.c` gründlich an. Dort finden Sie alles, was Sie für die Aufgabe brauchen

4. 1 Aufgabe

- Schreiben Sie ein Kernelmodul, das die geforderte Funktionalität realisiert.
- Das Modul *muss* aus *einer* C-Datei (und natürlich einer Header-Datei) bestehen.
- Das Modul *muss* `translate` heißen, das Kernel-Objectfile *muss* `translate.ko` heißen
- Das Modul *muss* seine Major-Nummer dynamisch vom Kernel erhalten.

Mit anderen Worten: Sie müssen die benötigte Funktionalität aus den Quellcodefiles `main.c` und `pipe.c` zusammenklauben und in eine Datei stopfen, und alle Erinnerungen an *scull* und *pipe* beseitigen. Außerdem müssen Sie das Makefile anpassen.

- Es *darf* in Ihrem Programm natürlich *nirgends* die magische Zahl 26 als Alphabetlänge vorkommen.
- Wer glaubt, für die String-Indexberechnungen eine ASCII-Tabelle benutzen zu müssen, hat in der Vorlesung, in der ich die Aufgabe erläutert habe (und vermutlich nicht nur in dieser) gepennt.

4. 2 Installation

Zur Insatallation des Moduls schreiben Sie zur Schonung Ihrer eigenen Nerven ein Skript, das folgendes tut:

- Alte Device-Nodes `/dev/translate?` entfernen.
- Altes Kernel-Modul entfernen.
- Neues Kernel-Modul laden.
- Major-Devicenummer aus `/proc/devices` mit `grep` und `cut` ermitteln.
- Neue Device-Nodes `/dev/translate0` und `/dev/translate1` mit der korrekten Major-Nummer erstellen.

4. 3 Hinweise

- Gehen Sie großzügig mit Debug-Meldungen per `PDEBUG` um.
- Erproben Sie die Zeichenumsetzungsalgorithmen in beiden Richtungen zunächst in einem Userspace-Programm, das Debuggen ist da viel einfacher.
- Das Headerfile für die *Kernel*-Stringfunktionen heißt `linux/string.h`
- Sie müssen sicherstellen, dass Ihr Substitutionsstring mindestens zwei mal so lang wie das Alphabet ist, sonst greifen Sie bei der Vorwärtsumsetzung ins Leere. Deshalb kopieren Sie die Modulparameter-Variable `translate_subst` mit der Funktion `strcpy()` auf die (statische globale) Arbeitsvariable `subst` und hängen ggf. Zeichen aus dem Vorgabestring an.

4. 4 Test / Abnahme

Folgende Dinge werde ich bei der Abnahme testen:

- Automatische Installation von Kernelmodul und Device-Nodes.
- Schreiben mit `echo` und Eingabeumleitung auf das Device, dann lesen mit `cat`.
- Beobachten der Debug-Meldungen im Kernel-Log.
- Erst Lesen mit `cat` starten, danach schreiben.
- Prüfen, ob der gleichzeitige Zugriff verhindert wird.
- Prüfen, ob die Kodierung und Dekodierung korrekt funktioniert.
- Laden des Moduls mit Parametern.

```
sudo insmod translate.ko \
  translate_subst="zyxwvutsrqponmlkjihgfedcbaZYXWVUTSRQPONMLKJIHGFEDCBA"
echo 'rvo Viulot!' > /dev/translate1
cat /dev/translate1          # ?????
```