4 Kryptographie à la Caesar

Schreiben Sie einen Gerätetreiber für zwei Geräte `/dev/caesare` und `/dev/caesard`, die das Ver- und Entschlüsseln von Text nach der Methode von G.I.Caesar durchführen. Die Methode besteht darin, jeden Buchstaben des Klartextes um eine feste Zahl von Stellen im Alphabet zyklisch zu verschieben. 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.

4. 1 Funktion

- Zum *Verschlüsseln* schreibt man Klartext auf `/dev/caesare`, und erhält beim Lesen von diesem Device den verschlüsselten Text. Zum *Entschlüsseln* schreibt man den verschlüsselten Text auf `/dev/caesard`, 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 *Verschlüsseln*
  Minor number 1 bedeutet *Entschlüsseln*.
- 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. Die Länge der zu verarbeitenden Texte soll natürlich unbegrenzt sein und insbesondere nicht von der Größe des Pufferspeichers abhängen.
- 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 und die Anzahl der Stellen, um die verschoben wird, sollen als Modulparameter festgelegt werden können. Beim Laden ohne Parameter sollen Standardwerte genommen werden.
- Alles was Sie zur Bearbeitung der Aufgabe wissen müssen steht in Kapitel 6, Abschnitt „Blocking I/O“ des Buches
  „Linux Device Drivers“ von Jonathan Corbet, Alessandro Rubini, und Greg Kroah-Hartman, die URL ist `http://oreilly.com/catalog/linuxdrive3/book/ch06.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.

4. 2 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* `caesar` heißen, das Kernel-Objectfile *muss* `caesar.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.

4. 3 Installation

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

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

4. 4 Hinweise

- Gehen Sie großzügig mit Debug-Meldungen per `PDEBUG` um.

4. 5 Test / Abnahme

Folgende Dinge werde ich bei der Abnahme testen:

- Automatische Installation des Moduls und Erzeugung der Device-Nodes
- Schreiben mit `echo` und Eingabeumleitung auf das Device, dann lesen mit `cat`.
- 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.

Dyh Fdhvdu – oder eher „glh vslqqhq, glh Urhphu“???
Jedenfalls viel Erfolg!