der Bundeswehr
Universität München

Universität der Bundeswehr
Fakultät für Elektrotechnik und Technische Informatik
ETTI – Institut 2
Prof. Dr. H. Görl

HT 21
Praktikumsblatt 3
9. November 2021

Praktikum zu
Sicherheit moderner Betriebssysteme

Aufgabe 1 (Kernel-Treiber: `char`-Gerätetreiber)

Die wichtigste Klasse von Unix-Gerätetreibern sind die sog. `Character-Driver` oder zeichenorientierten Gerätetreiber. In dieser Aufgabe soll ein solcher Treiber erstellt werden, der zur Untersuchung des Kernadressraums dienen wird.

(a) Wechseln Sie in das `/dev`-Verzeichnis und sehen Sie sich die Gerätedateien mit `ls -l` dort an. Welche Geräteklasse (Zeichen „c“ oder „b“ ganz links dargestellt), Berechtigungen (rwx) und Dateigröße (in Bytes) erhalten Sie? Beachten Sie hauptsächlich die Treiber mit der Typbezeichnung `c`.

(b) sehen Sie sich nun die Gerätedateien mit `ls -ln` an. An der Stelle, an der sonst Eigentümer und Gruppenzugehörigkeiten einer Datei angezeigt werden, stehen bei Gerätetreibern die sog. Major- und Minor-Nummern eines Treibers, die zur Kommunikation mit dem Treiber, analog zu Speicheradressen, genutzt werden.

(c) Suchen Sie in der Prozessliste (angezeigt mit `ps -axj`) den Prozess Ihrer aktuell genutzten Shell (typischerweise „bash“). Finden Sie den Namen des Terminalgerätes aus der Prozessliste (z.B. `pts/0`) heraus. Suchen Sie den entsprechenden Treibereintrag im Verzeichnis `/dev/pts/` und geben Sie einen beliebigen Text auf diesem Terminal mittels `echo „TEXT\ > /dev/pts/0` aus. Schreiben Sie ein C-Programm, das auf allen aktiven Terminals einen Hinweis (quasi) gleichzeitig ausgibt.

(d) Erzeugen Sie eine Gerätedatei für ein aktives Terminal nun mit folgendem Kommando in einem beliebigen Verzeichnis, vorzugsweise in Ihrem Homeverzeichnis: (MAJOR und MINOR sind die entsprechenden Nummern, die Sie mit `ls -l` im Geräteverzeichnis sehen)

Hinweis: eine freie Majornummer finden Sie, indem Sie die Datei `/proc/devices` ansehen, die die aktuell belegten Majornummer anzeigt.

```
1    mknod GERAETENAME c MAJOR MINOR
```

(e) Laden Sie einen der Treiber eines vergangenen Versuchs in den Kernel. Versuchen Sie den Inhalt der Datei `/proc/kallsyms` zu verstehen. Wozu dienen die angegebenen Adressen? Suchen Sie die Einträge der geladenen Gerätetreiber. Welche Einträge finden Sie und was bedeuten dabei die angegebenen Adressen?

(f) Schreiben Sie einen Char-Treiber als Modul für den Linux-Kernel, dass ein Benutzerprogramm in das Gerät einen Satz hineinschreiben kann und anschließend dieser Satz mittels `cat DATEI` wieder aus der Gerätedatei ausgelesen werden kann.

(g) Ändern Sie den Treiber nun derart ab, dass der Benutzer eine Adresse in hexadezimaler Form in die Gerätedatei eingeben kann (z.B. `echo "fed00000» CHARDATEI`) und ein Speicherbereich von fester Länge (z.B. 256 Byte) als Hexdump von dieser Adresse beginnend ausgelesen wird. Damit kann später der Kernadressbereich durch einen Benutzerprozess zur Laufzeit untersucht werden.

Aufgabe 2 (Ein erster startbarer Kern)

In dieser Aufgabe wird ein erster einfacher Kernel zusammengebaut, mit einem Bootlader startfähig gemacht und getestet. Dazu finden Sie auf der Praktikumsseite eine Archivdatei `os.zip`, in der ein Grundgerüst für einen Betriebssystemkern zusammengestellt ist. Falls Sie sich über derartige Projekte näher informieren möchten, sehen Sie sich bitte die Seite im Netz „osdev“ an. Dort sind viele kleine Betriebssystemprojekte verlinkt und auch hervorragend erklärt, wie man erste startbare Kerne erzeugt.

(a) Laden Sie die Archivdatei `os.zip` herunter, entpacken Sie diese und sehen Sie sich zunächst die einfache Datei `build.sh` zur Erzeugung des Kerns an. Schreiben Sie ein Makefile, das zwei Ziele kennt, „build“ und „clean“ jeweils zum Erzeugen des Kerns und zum Löschen der Objektdateien und des Kerns.

(b) Starten Sie die mitgelieferte Floppy-Disk Image-Datei mit dem Virtualisierungsemulator `qemu`, den Sie bitte ggf. nachinstallieren. Informieren Sie sich dazu zunächst darüber, wie man auf der Kommandozeile dem Emulator `qemu` beibringen kann, dass eine Imagedatei gestartet werden soll. Hinweis: dazu ist ein Schalter nötig.

(c) Übersetzen Sie die Quellen mit Ihrem Makefile. Das erzeugte File `kernel.bin` ist der selbst erzeugte Betriebssystemkern, der von einem Bootlader, hier dem `grub` gestartet

werden muss. Diese Kerndatei muss nun in die Floppy-Disk Image-Datei eingebaut werden. Dazu führen Sie als Benutzer `root` folgende Schritte durch:

i. Erzeugen Sie ein temporäres Verzeichnis (z.B. mit dem Namen `fdtmp`), in welches Sie den Inhalt dieser Floppy-Disk Image-Datei abbilden.

   Hintergrundinformation: Es ist unter Linux leicht möglich, eine Image-Datei wie ein übliches Blockgerät in das Verzeichnissystem einzuhängen und für Leseoperationen zu benutzen. Dazu gibt es einen eigenen Blockgerätetreiber im Kernel, der `Loopback-Driver`.

ii. Bilden Sie den Inhalt der Image-Datei durch folgendes Kommando in das neue Verzeichnis ab (mounten):

```
1       mount -t loop grub+kernel.bin.img fdtmp
```

   Wenn Sie anschließend in das Verzeichnis fdtmp wechseln, können Sie den Inhalt der Datei ansehen und verändern.

iii. Kopieren Sie Ihren neuen Kern in das Verzeichnis `fdtmp` über den alten Kern `kernel.bin`. Dadurch ist der neu erzeugte Kern im Floppy-Disk-Image enthalten.

iv. Verlassen Sie das Verzeichnis (z.B. durch das Kommando „`cd ..`“ und entfernen Sie die Abbildung der Image-Datei in das Verzeichnis `fdtmp` durch folgendes Kommando:

```
1       umount fdtmp
```

   Starten Sie nun die veränderte Image-Datei mit dem Virtualisierungswerkzeug `qemu` analog der zweiten Teilaufgabe.

(d) Können Sie anhand der Image-Datei, etwa deren Zeitstempels, feststellen, dass der Inhalt (z.B. der enthaltene Kern verändert worden ist?

(e) Erweitern Sie die Funktion `main()` um Ausgaben, die den jeweiligen Startzustand anzeigen, also etwa „Starte Initialisierung der GDT...“.

(f) An welcher Stelle wird vom Assembler-Teil in `start.asm` in den C-Teil gewechselt. Wie funktioniert der Aufruf von Unterbrechungen, die ja im C-Teil in der Funktion `irq_handler` behandelt werden und wie werden Argumente bzw. die Unterbrechungsnummer übergeben? Wie funktioniert die Auswahl der korrekten Unterbrechungsbehandlungsroutine?

(g) In PC-Architekturen ist ein eigener Baustein, der sog. PIC (Programmable Interrupt Controller) u.a. für die Erzeugung periodischer Signale zuständig. Dieser signalisiert über den IRQ0 mit einer Frequenz von 18,222 Hz. Eine eingerichtete Signalhandler-Funktion auf dieser Unterbrechung wird demnach ca. 18 mal pro Sekunde aufgerufen. Implementieren Sie die Funktion `timer_handler()`, die ca. alle 2 Sekunden einen beliebigen Text auf dem Bildschirm ausgeben soll.

Hinweis 1: Benutzen Sie dafür bitte die globale Variable `timer_ticks` in der Datei `timer.h` als stetig fortlaufende Zählvariable durch den Signalhandler hochgezählt wird.

Hinweis 2: Die Zahl 18,222 Hz wurde von den PC-Entwicklern Ende der siebziger Jahre festgelegt und wird heute noch durch die BIOS-Funktionen gesetzt. Es gibt vermutlich drei Erklärungen für die Festlegung auf diese relativ unpraktische Zahl: 1. Der Schrittmotor von Floppy-Disk-Laufwerken musste mit dieser Frequenz gesteuert werden. Oder 2. Wenn ein 16-Bit großer Zähler verwendet wird, dann tritt der Überlauf genau nach einer Stunde ein.

(h) Realisieren Sie zudem eine Funktion `timer_wait(int ticks)`, die für eine übergebene Anzahl an Ticks wartet und erst nach Ablauf dieser Zeit zurückkehrt.

(i) Erweitern Sie Ihr Makefile um das automatische Einbauen des erzeugten Kerns in die Floppy-Disk Image-Datei.

(j) Implementieren Sie die Funktionen der vier Cursortasten „nach oben“, „nach unten“, „nach links“ und „nach rechts“. Sehen Sie dazu in die Dateien `screen.c` und `keyboard.c`.

(k) Realisieren Sie die Funktion `memcpy()` mit folgender Funktionssignatur:

```
1    void *memcpy(void *dest, const void *src, size_t count)
```

Die Funktion soll eine Anzahl `count` an Zeichen aus dem Speicherbereich beginnend mit der Adresse `src` an die Adresse beginnend mit `dest` kopieren.

Testen Sie diese Funktion `memcpy()`, indem Sie einen Bildschirmbereich an eine andere Stelle des Bildschirms kopieren. Hinweis: Sehen Sie sich dazu die Datei `screen.c` an.