4. Übung aus Treiberentwicklung SS2015, Abgabetermin: 10.6.2015

Ausgearbeitet von: Gruppe:

Aufgabe 1: Portierung des I²C-Interface-Treibers nach Linux

Der in der vorigen Übung erstellte Device-Treiber soll nun nach Linux portiert werden. Das Treibermodell der Betriebssysteme Windows und Linux hat gewisse Ähnlichkeiten:

- Abstraktion als File
- Lesen+Schreiben von/auf Files
- IOCTL-Kommandos

Von der Philosophie her gibt es einen großen Unterschied:
Unter Windows wird das API zum Treiber hin nicht verändert. Bei Linux hingegen wird das API optimiert wenn es notwendig erscheint. Der Vorteil bei Windows besteht darin, dass Treiber-Sourcen auch nach vielen Jahren noch ohne Änderungen übersetzt werden können (siehe wdm1). Der Vorteil bei Linux besteht darin, dass Fehler/Unvollständigkeiten/etc. bei der API-Definition ausgebessert werden.

Unter `http://linux.die.net/lkmpg/x892.html` bzw. in `linux-ioctl-example.html` finden Sie ein kompaktes Beispiel für einen Linux-Treiber der das IOCTL-Interface verwendet. Dieses Beispiel enthält sowohl die Kernel-Mode-Komponente als auch ein User-Mode-Programm das mit dem Treiber über ioctl-Kommandos interagiert.

`http://linux.die.net/lkmpg/x181.html` bzw. `compiling-linux-kernel-modules.html` zeigt, wie ein Kernel-Module kompiliert werden kann.

Anstelle des Debug-Print-Drivers kann die Funktion `printk` verwendet werden. Dieser Output kann mit dem Befehl dmesg angezeigt werden. Mit Linux-Boardmitteln kann der Output von dmesg ständig überwacht werden: `watch -n1 'dmesg | tail -50'`

Vorgehensweise:

- Sie können auf obigem Linux-Treiber aufsetzen um das I²C-Protokoll des Windows-Treibers zu integrieren (die API dieses Treibers ist nicht mehr auf dem aktuellen Stand. Eine Google-Suche nach Fehlermeldungen ist meist zielführend.)
- Die User-Mode Applikation soll wieder in Python realisiert werden. Dazu muss die Klasse HWDevice aus `wdm1-test.py` so erweitert werden, dass sie sowohl unter Windows als auch unter Linux lauffähig ist (sys.platform gibt Auskunft auf welchem System das Script läuft). Dazu muss vor allem OpenDrv, CloseDrv und DeviceIoControl für Linux implementiert werden. Wichtig ist bei DeviceIoControl vor allem, dass die numerischen IOCTL-Werte vom Aufruf mit denen im Device-Treiber übereinstimmen.
- Dann müssen Sie herausfinden, welche Funktion unter Linux anstelle von `WRITE_PORT_UCHAR` bzw. von `READ_PORT_UCHAR` eingesetzt werden muss.

Abzugeben sind:

- C- und Python-Sourcefiles
- Ausgaben des Testprogramms
- Ausgaben des Treibers
- Beantwortung folgender Fragen:
  a) Welche Probleme traten bei der Portierung des Treibers auf?
  b) Welche Vorkehrungen können getroffen werden damit der Treiber einfacher portiert werden kann?