1. Übung aus Treiberentwicklung SS2015, Abgabetermin: 15.4.2015

Ausgearbeitet von: Gruppe:

Allgemeines zur Übung:

Ziel der Übung ist, dass Sie sich in ein komplexes System einarbeiten können, um danach eine Erweiterung dieses Systems durchführen zu können. Bei diesem komplexen System handelt es sich um einen Windows Device Driver.

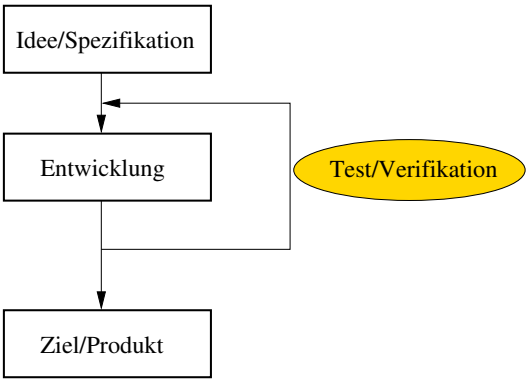

Abbildung 1: Entwicklungsablauf

Die Abbildung 1 zeigt die Entwicklung eines Produktes von der Spezifikation ausgehend. Während der Entwicklungsphase sind viele Iterationen notwendig. Um möglichst schnell zu einem funktionstüchtigen Produkt zu kommen sind zwei Punkte wesentlich:

- Kurze Iterationen
- Wenige Iterationen

Diese beiden Ziele sind durch eine interaktive Testumgebung mit hohem Abstraktionsgrad erreichbar. Dazu wird ein funktionales Verifikationskonzept mit der Programmiersprache Python (`http://www.python.org`) in den nächsten Übungsstunden vorgestellt.
Python ist einfach zu erlernen und ermöglicht eine schnelle Programmerstellung. Die Programmiersprache wird auch bereits stark im industriellen Umfeld eingesetzt. Als Beispiel sei Google angeführt (weitere Success Stories unter: `http://www.python.org/about/success/`). Ein interessanter Vergleich über die Verbreitung von Programmiersprachen ist der Tiobe-Index (`http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`).
Die einzelnen Übungen sind bis zum Abgabetermin schriftlich auszuarbeiten. Pro Übungsgruppe muss eine Ausarbeitung abgegeben werden (**Hinweis:** alle Gruppenmitglieder müssen an der Ausarbeitung mitarbeiten und diese auch gegebenenfalls erklären können).

Aufgabe 1: Analyse eines „einfachen“ Treibers

Basierend auf dem Treiber WDM1 von Chris Cant sollten Sie sich ein erstes Bild über die Treiberentwicklung unter Windows machen. Basierend auf diesem Treiber werden Sie schrittweise eigene Funktionalität hinzufügen.
Der WDM1-Treiber zeigt, wie vom User-Mode auf den Kernel-Mode zugegriffen werden kann. Zudem wird der Debug-Print-Treiber eingesetzt, um die Vorgänge beim Aufruf des Device-Treibers nachvollziehbar zu machen.

Beispiel 1.1: Treiber übersetzen/verwenden

In der Übung sollten Sie folgende Punkte herausfinden und in Ihrer Ausarbeitung beschreiben:

- Wie wird der Treiber übersetzt?
- Wie wird der Treiber installiert?
- Wie kann man von einer User-Mode-Applikation auf den Treiber zugreifen?

Beispiel 1.2: Übersichtsgrafik

Als nächstes erstellen Sie einen grafischen Überblick über zumindest folgende Dateien/Komponenten:

- Wdm1Test.cpp
- Pnp.cpp
- Ioctl.h
- Init.cpp
- GUIDs.h
- Dispatch.cpp
- DebugPrint.h
- DebugPrint.c
- Wdm1.sys
- Wdm1free.inf
- Wdm1checked.inf
- DebugPrintMonitor.exe
- DebugPrt.sys

Die Übersichtsgrafik soll auf jeden Fall zeigen welche Teile im User-Mode bzw. im Kernel-Mode liegen. Zudem soll über Pfeile dargestellt werden, wie einzelne Komponenten miteinander in Beziehung stehen. Die Grafik soll z.B. als Grundlage dienen können, um jemandem die Struktur des Treibers erklären zu können.

Beispiel 1.3: Erklärung der Übersichtsgrafik

Erklären Sie die erstellte Grafik mit eigenen Worten. (**Hinweis:** Die Übungen werden miteinander verglichen.)

Beispiel 1.4: Informationsquellen

Welche Informationsquellen haben Sie zur Lösung dieser Aufgabe herangezogen?

Aufgabe 2: Schlussfolgerungen

- Welche Stolpersteine sind Ihnen bis jetzt aufgefallen?
- Einschätzung der weiteren Vorgehensweise: Bei welchen Teilen herrscht noch Unklarheit? Wie werden Sie weiter vorgehen, um bessere Klarheit über den Beispieltreiber zu erlangen?

Testen des Treibers mit Python:

Einsatz von Python zum interaktiven Test eines Device Drivers, erste Änderungen an einem Device Driver um die Schnittstelle zwischen User-Mode und Kernel-Mode zu verwenden.

Der Einsatz von Python zum Testen von Device-Treibern oder anderer C-Software hat mehrere große Vorteile:

- Kein Kompilieren notwendig
- Interaktive Eingabe von Kommandos möglich (z.B. im Emacs)
- Highlevel Datentypen stehen zur Verfügung (Listen, Files, Strings, etc.)
- Es steht eine große (ständig wachsende) Standardbibliothek (mit guter Dokumentation) zur Verfügung
- Direktes Interagieren mit Device-Treiber ist ähnlich einer Unix-Shell möglich. Die Verwendung von Prozeduren erlaubt den Zusammenbau komplexer Tests aus kleinen Komponenten.

Aufgabe 3: Schnelleinstieg in Python

Zu Python gibt es eine Menge interessanter Ressourcen im Internet. Wir werden Python einsetzen um einfache Unterprogramme zu schreiben. Weiters ist es wichtig herauszufinden, wie If-Abfragen oder Schleifen beschrieben werden. Eine sehr kompakte Einführung zu Python ist unter `http://hetland.org/writing/instant-python.html` zu finden.
Bitte lesen Sie das dazu die obige Einführung und geben Sie Beispiele zu folgenden Konzepten an (die Beispiele aus experimentier-kasten.py sowie instant-python.html gelten nicht):

- Funktionsdefinition und Funktionsaufruf
- Eine For-Schleife
- Eine If-Abfrage
- Die Verwendung eines Arrays

- Die Verwendung eines Dictionaries
- Definition einer Klasse, anlegen einer Instanz dieser Klasse und Aufruf einer Methode

Zum Austesten der geforderten Programmteile kann z.B. Emacs oder IPython verwendet werden.
Hinweis: der Beispiel-Code soll auch Teil der Übungsabgabe sein.

Aufgabe 4: Test des Treibers mit Python

Das Python Script `wdm1-test.py` ermöglicht es, den WDM1-Treiber mittels der Skriptsprache Python zu Testen. Dazu ist eine Python-Installation notwendig, die auch die Python Win32-Extensions enthält (Paket pywin32).
Die Python Win32-Extensions ermöglichen den Aufruf von Win32-API-Calls von Python aus. Der Großteil der benötigten Funktionen für das Testen von Device Treibern ist in dieser Bibliothek bereits verfügbar. Lediglich die spezielle Funktion `GetDeviceViaInterface` ist leider nicht verfügbar. Um diese Funktion einsetzen zu können, ist es notwendig, eine Python-Erweiterungs-DLL (`DeviceDriverAccess.dll`) zu erstellen. Diese DLL wird Ihnen bereits vorkompiliert zur Verfügung gestellt (sie ist für Python2.5 kompiliert).
Eine Erweiterung des Skripts `wdm1-test.py` um Aufrufe von `SetFilePointer`, und `DeviceIoControl` ermöglicht eine ähnliche Testfunktionalität wie mit dem bisherigen Test mittles `Wdm1Test.cpp`.
Schreiben Sie ein Testprogramm das `Read`, `Write`, `SetFilePointer` und `DeviceIoControl` verwendet um mit dem WDM1-Treiber zu interagieren. Verifizieren Sie mittels der Rückgabewerte des Treibers und dem DebugPrintMonitor-Output ob die Aufrufe korrekt sind.

Hinweise:

- Folgende Vorlage kann verwendet werden, um die Methode `DeviceIoControl` zu realisieren

```
FILE_DEVICE_UNKNOWN = 0x00000022

METHOD_BUFFERED = 0
METHOD_IN_DIRECT = 1
METHOD_OUT_DIRECT = 2
METHOD_NEITHER = 3
FILE_ANY_ACCESS = 0

def CTL_CODE(DeviceType, Function, Method, Access):
   return (DeviceType << 16) | (Access << 14) | (Function << 2) | Method

IOCTL_USB_GET_DEVICE_DESCRIPTOR = CTL_CODE(FILE_DEVICE_UNKNOWN, 0x802, METHOD_BUFFERED, FILE_ANY_ACCESS)

win32file.DeviceIoControl(drvHnd, IOCTL_USB_GET_DEVICE_DESCRIPTOR, "", 512)
```

- Das Pythonmodule struct bietet die Funktionen unpack sowie pack an. Um die BufferSize als Integer-Wert zu erhalten sollten Sie die Funktion unpack einsetzen.