

Prof. Dr. Chr. Vogt

von Kulyk Nazar

(Matr.Nr.:258360030100)

Ich habe die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet hat.

Aufgabe 1.

>Schauen Sie nun nach, welche Zahlen in der Datei stehen (nicht notwendig vom Programm aus). Wahrscheinlich sieht es so aus, als ob die asynchronen I/Os keine Zeit brauchen.

Im erstem Fall, brauchen alle I/Os immer keine Zeit. Und es werden immer „0“ in die Datei geschrieben. Allerdings gibt es unterschied ob die Datei schon existierte, oder nicht.

Falls die Datei nicht existierte bekommt man IO_PENDING Fehler mit hilfe von GetLastError. Die Datei wird asynchron geschrieben, aber wegen dem NTFS File Caching dauern alle I/Os keine Zeit.

Im Fall wo die Datei schon bereits existierte macht NTFS alle I/Os synchron, auch wenn es einen asynchronen Zugriff angefordert wurde.

Aufgabe 2.

>Versuchen Sie, ob sich das Verhalten durch Ausschalten des Write-Back-Caching ändert.

Wenn man die Datei Offnet (CreateFile()) mit dem Flag FILE_FLAG_WRITE_THROUGH, werden alle I/Os ohne Caching ausgeführt. Man hört sogar wie die Festplatte arbeitet werend das Programm wird ausgeführt.

Man kann im File dann ablesen wie lange dauern die I/Os. Ab dem Byte 729 brauchen alle weiteren I/Os Zeit.

```
Lister - [c:\test.dat]
File Edit Options Help 2 %
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
150
0
0
0
0
0
0
0
0
0
0
342
0
2347
1192
1053
1151
1127
1106
992
1094
1108
973
1841
1171
```

Aufgabe 3.

Wie ich schon in Aufgabe 1 geschrieben habe, werden alle I/Os synchron ausgeführt falls die Datei schon existierte.

Das ist etwas gefährlich, man soll es immer im Kopf behalten, falls man im Programm Asynchrone I/Os braucht.

Aufgabe 4.

Mit Hilfe von nfi kann man MFT Sektoren raus finden wo die Information über die auf NTFS Partition legenden Dateien und wie die fragmentiert sind.

```
C:\WINDOWS\system32\cmd.exe
D:\prog\asyncio1\tools>nfi c:
NTFS File Sector Information Utility.
Copyright (C) Microsoft Corporation 1999. All rights reserved.

File 0
Master File Table ($Mft)
  $STANDARD_INFORMATION (resident)
  $FILE_NAME (resident)
  $OBJECT_ID (resident)
  $DATA (nonresident)
    logical sectors 6291456-6369983 (0x6000000-0x6132bf)
    logical sectors 38281216-38288895 (0x2482000-0x2483dff)
  $BITMAP (nonresident)
    logical sectors 6291448-6291455 (0x5ffff8-0x5fffff)
    logical sectors 50704472-50704479 (0x305b058-0x305b05f)
```

Jetzt schauen wir wie sehen die Informationen für von unserem Programm erzeugter Datei:

```
C:\WINDOWS\system32\cmd.exe
NTFS File Sector Information Utility.
Copyright (C) Microsoft Corporation 1999. All rights reserved.

\test.dat
  $STANDARD_INFORMATION (resident)
  $FILE_NAME (resident)
  $DATA (nonresident)
    logical sectors 124576-124583 (0x1e6a0-0x1e6a7)

D:\prog\asyncio1\tools>
```

Hier können wir raus lesen das die Datei ist in einem Segment (nicht fragmentiert) nicht resident auf der Platte unter Sektoren von 124576 bis 124583.

Jetzt Schreiben wir nur weniger als 512 Byte.

```
C:\WINDOWS\system32\cmd.exe
D:\prog\asyncio1\tools>nfi c:\test.dat
NTFS File Sector Information Utility.
Copyright (C) Microsoft Corporation 1999. All rights reserved.

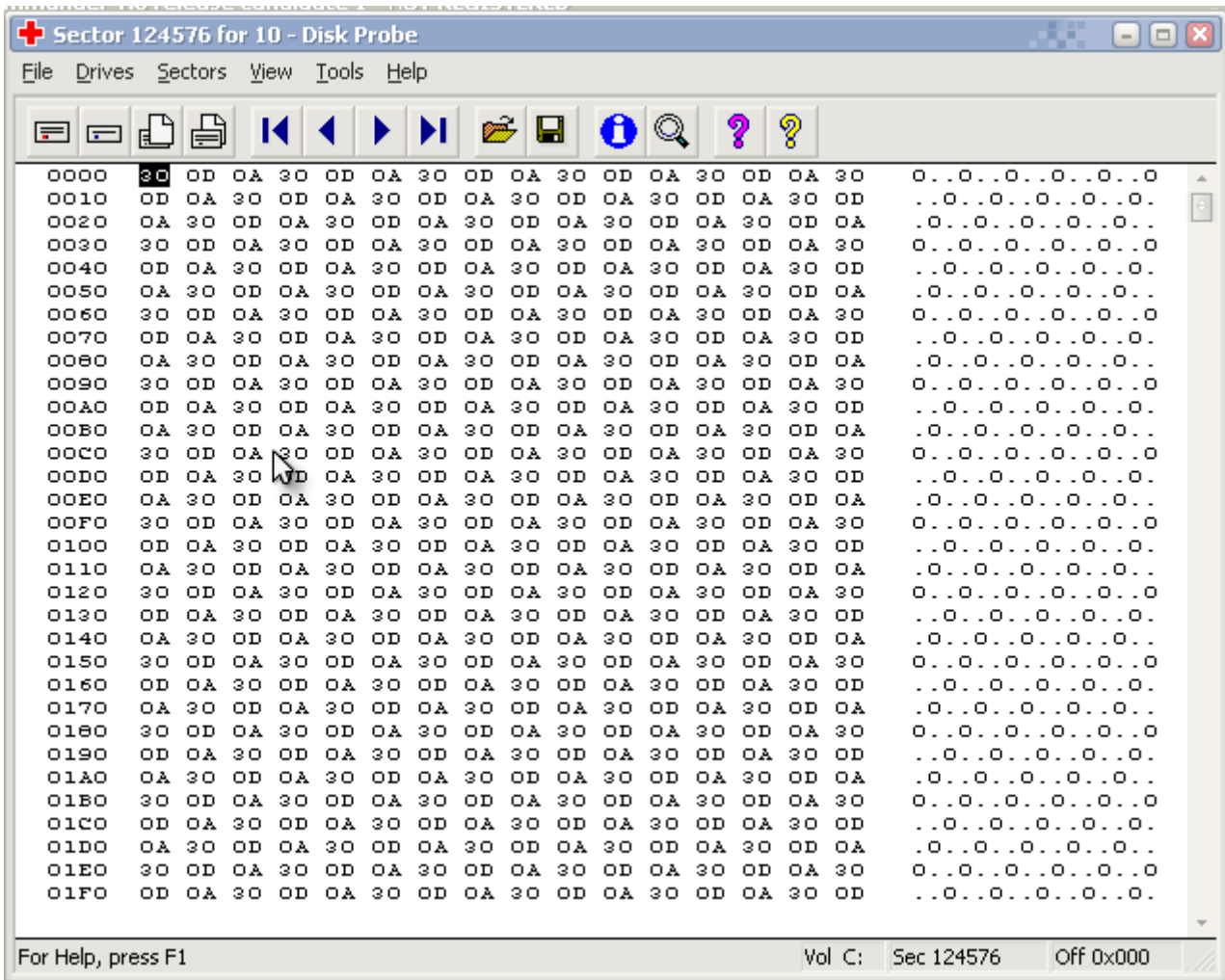
\test.dat
  $STANDARD_INFORMATION (resident)
  $FILE_NAME (resident)
  $DATA (resident)

D:\prog\asyncio1\tools>
```

Hier ist das unterschied sichtlich. Es werden keine Daten auf die Platte geschrieben.

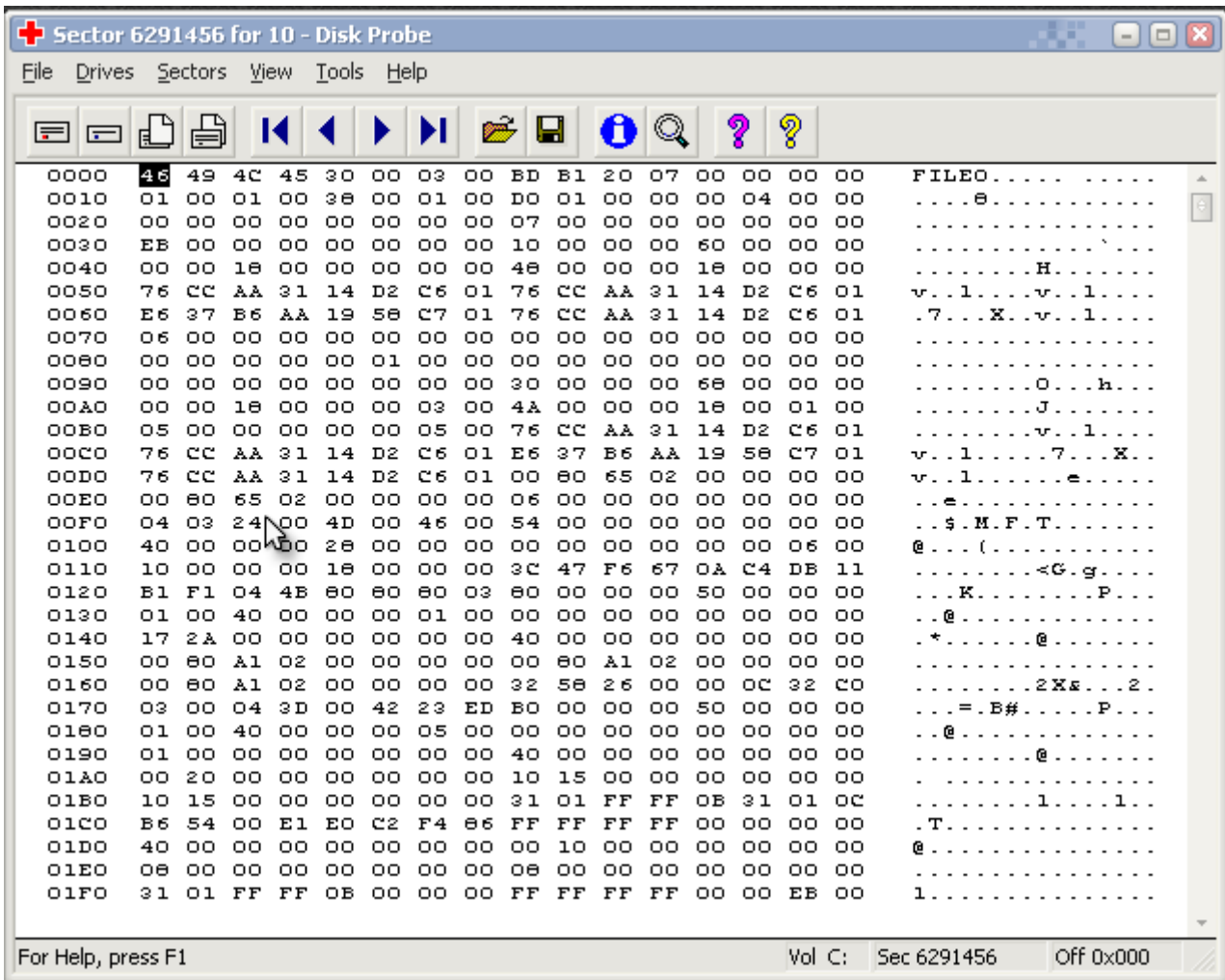
Also solange unseres schreiben Vorhang nicht die 512 Byte Grenze erreicht, schreib NTFS die Datei nicht auf die Platte.

Und jetzt schauen wir die Datei die wir mit 1024 Byte schreiben nun Direkt an:



Hier können wir sehen unsere 0+r\n die wir aus dem Programm schreiben.

So sieht das MFT Eintrag auf der Platte:



nfi.exe tut nichts anderes als diese Daten auszuwerten für alle Dateien.