Informatik

Arbeit mit Dateien unter Visual Basic

Prof. Dr.-Ing. Thomas Wiedemann
Fachgebiet Informatik / Mathematik

Begriff Datei (engl. File)

Begriff :

- Zusammenfassung von Daten, welche in der Regel auf nichtflüchtigen Massenspeichern abgelegt und verwaltet werden
- spezielle Verfahren zum Finden und Verarbeiten:
  - Verzeichnisstrukturen (mit schnellem Hash-Zugriff)
  - FAT (File Allocation Table) zur Reservierung der Daten auf der Festplatte

Nach der Art des Inhaltes werden unterschieden :

- **Binärdateien**
  - enthalten direkt die Bitmuster der Daten und können in der Regel nicht durch allgemeine Programme gelesen werden
  - die meisten Programmdaten (*.doc, *.xls, *.cad ..) sind Binärdateien
- **Textdateien**
  - enthalten nur reinen Text (und einfache Steuerzeichen)
  - viele Steuerdateien und die Internetformate sind Textdateien (*.htm , *.xml)
  - Die Verarbeitung von Textdateien ist etwas langsamer, jedoch deutlich flexibler !

Gegenwärtige Tendenz :

- Während in der Vergangenheit die meisten Dateien als Binärdateien abgelegt wurden, werden zukünftig die meisten Daten als Textdateien gespeichert !

Technische Realisierung der Speicherung

Begriff Massenspeicher:

- Technische Geräte mit einer im Vergleich zum Arbeitsspeicher sehr großen Speicherkapazität, in der Regel mit dauerhafter Speicherung (>= 10 Jahre)

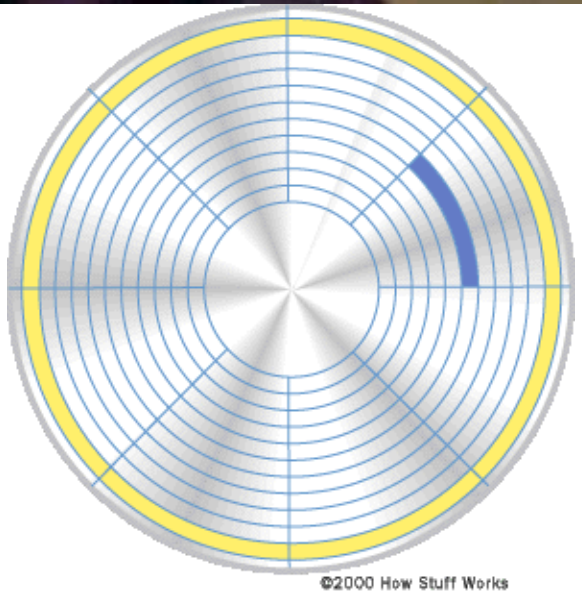
Nach der Art der Speicherung werden unterschieden :

- **Magnetische Verfahren (Details siehe www.howstuffworks.com)**
  - Magnetische Materialien werden unterschiedlich magnetisiert und speichern die Information durch die Orientierung des Magnetfelder (Nord-Südpol)
  - Magnetbänder (veraltet)
  - Disketten , ZIP-Disketten
  - Festplatten
- **Optische Verfahren**
  - durch unterschiedliche geometrische Strukturen oder chemische Stoffeigenschaften wird das Reflexionsvermögen von Oberflächen verändert und meist mittels stark fokussierter Laserstrahlen ausgelesen
  - CD-ROM, DVD : „Tiefdruck“ von Bitmustern (Pit und Lands)
  - CD-RW, DVD-RW : Phasenveränderung des Materials durch Laserstrahl
- **Elektronische Verfahren**
  - USB-Stick : Speicherung von Ladungsträgern in sehr gut isolierten Bereichen auf Chips
- **Zukünftig ?** - Holografische Verfahren, Atom- oder Molekülllevel

Speicherung auf Festplatten

Aufbau Festplatte :

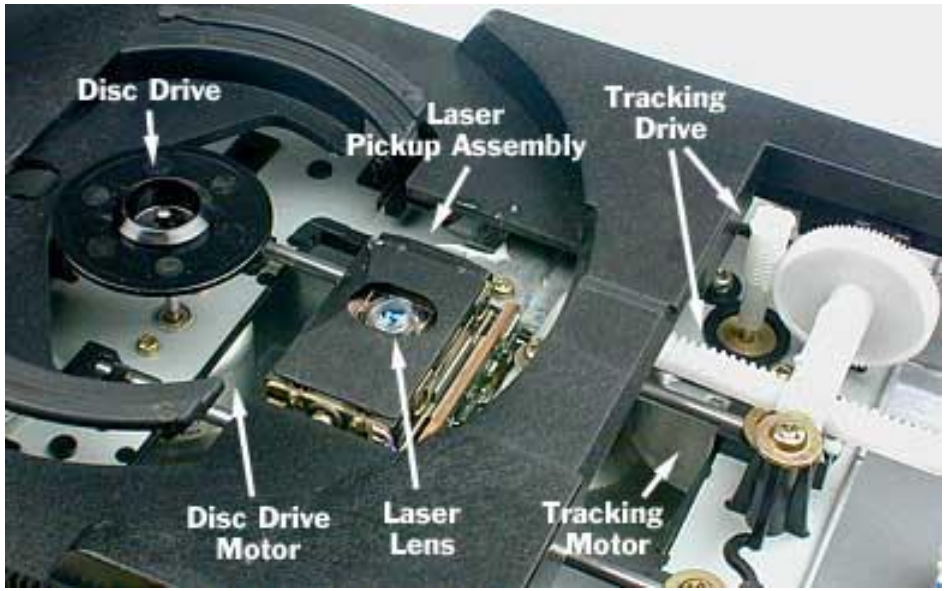
- **Platten mit magnetischer Oberflächenbeschichtung**
- **Platte darf wegen Staub NIEMALS geöffnet werden**
- **Lesekopf fährt konzentrische Zylinder ab (bei Disketten z.B. 80)**

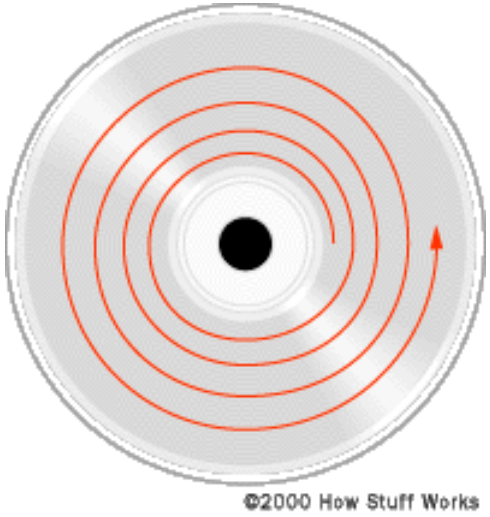


- **Platte ist insgesamt in Zylinder und Sektoren eingeteilt**
- **Anzahl und Aufteilung ist produktspezifisch – wird aber vom Betriebssystem auf eine große Anzahl von Sektoren heruntergerechnet**

Quelle Abb.: www.howstuffworks.com

Speicherung auf CD´s / DVD´s





Aufbau :

- **Siehe Abb.**
- **Im Gegensatz zur Festplatte gibt es nur eine SPUR , welche spiralförmig verläuft**
- **Gesamte Spur ist jedoch wieder in Sektoren eingeteilt**
- **Anzahl und Aufteilung ist produktspezifisch – wird aber vom Betriebssystem auf eine große Anzahl von Sektoren heruntergerechnet**

Quelle Abb.: www.howstuffworks.com

Allgemeine Grundoperationen auf Dateien

Öffnen der Datei

- Betriebssystem sucht die Datei
- bei Erfolg wird ein Puffer für die Daten der Datei angelegt
- ein Pointer (= Handle) auf den Puffer wird zurückgegeben

Lesen in der Datei

- Programm stellt über Handle (=ZugriffsID) eine Leseanfrage
- Betriebssystem liest in der Datei
- bei Erfolg werden im Puffer die Daten geliefert

Schreiben in der Datei

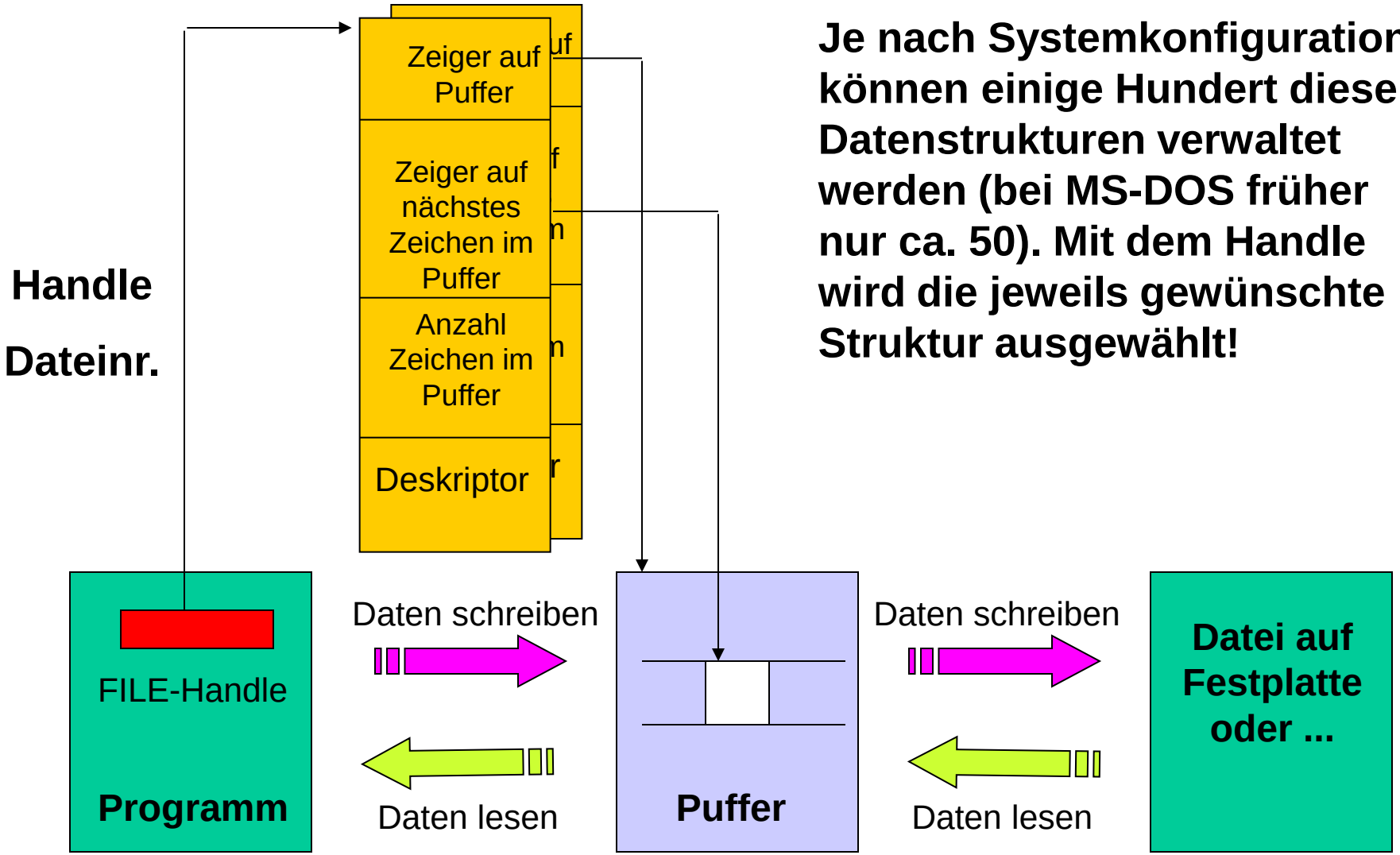
- Programm schreibt seine Daten in den Puffer
- stellt über Handle eine Schreibanfrage
- Betriebssystem schreibt sofort oder später in die Datei
- bei Erfolg wird Erfolgsmeldung über Handle geliefert

Schliessen der Datei

- Betriebssystem löscht den Puffer und entfernt eventuelle Sperren auf der Datei (wird das Schliessen vergessen, können andere Programme nicht auf die Datei schreiben !!)

Technische Realisierung des Zugriffs auf Dateien

Datei-Datenstruktur im Betriebssystem



Je nach Systemkonfiguration können einige Hundert dieser Datenstrukturen verwaltet werden (bei MS-DOS früher nur ca. 50). Mit dem Handle wird die jeweils gewünschte Struktur ausgewählt!

Funktionen zum Öffnen und Schließen von Dateien

- Der Pfad und der Dateiname werden als String gespeichert !
- Meist werden Pfad und Dateiname zusammengesetzt !
- Das Handle ist bei VB eine einfache ganze Zahl !

```
Dim Dateiname As String, Dateinr As Integer
Dateiname = "c:“  &   “autoexec.bat“
```

- Das Handle muß unmittelbar vor dem Öffnen durch die Funktion Freefile() ermittelt werden (nicht auf Verdacht anfordern!!!)

```
Dateinr = FreeFile: Rem Hole freies Handle
```

- für das Öffnen und Schließen von Dateien ist folgende Sequenz zu verwenden:

```
Open Dateiname For Input As Dateinr
   Rem Dateioperationen
Close Dateinr
```

Optionen beim Öffnen von Dateien (Auszug)

Open Dateiname For Input As Dateinr

- öffnet die Datei im **LESE-Modus (Schreiben ist NICHT möglich)**

Open Dateiname For Output As Dateinr

- öffnet die Datei im **Schreib-Modus**
- **Achtung 1: Der Inhalt der Datei wird auf 0 gesetzt !**
- **Achtung 2: Falls eine entsprechende Datei bereits existiert, wird deren Inhalt dadurch GELÖSCHT !!! -> DATENVERLUST !!**

Open Dateiname For Append As Dateinr

- öffnet die Datei im **Schreib-Modus** nach dem letzten Byte
- wesentlich sicherer als OUTPUT –> IDEAL für LOG-Dateien oder ähnliche Aufgaben (z.B. Messwert aller 10 min zu Liste hinzufügen)

Das zeilenweise Lesen von Dateien

- Nach einem erfolgreichem Öffnen einer Datei im Lesemodus kann diese mit Line Input #Dateinr, s sequentiell gelesen werden.
- Da die Dateilänge i.d.R. beliebig ist, muß das Lesen in einer Schleife bei ständigem Test auf das Dateiende durchgeführt werden.
- die Funktion eof(Dateinr) gibt **WAHR** beim Dateiende zurück

```
Dateinr = FreeFile: Rem Hole freies Handle
Dim  Zeilen(1000) as string
Open Dateiname For Input As Dateinr
      Do until EOF(Dateinr):    Rem Dateioperationen
        Line Input #Dateinr, s  :   Rem liest genau eine Zeile
        Debug.Print s           : Rem gibt diese testweise aus
        Zeilen( i ) =  s  :   i = i +1  :  Rem in Vektor ablegen ...
      Loop :  Rem und oben erneute auf Dateiende prüfen ...
Close Dateinr  : Rem Schließen NICHT vergessen !!
```

Das Einlesen von Variablen aus Dateien

- Neben dem Einlesen einer kompletten Zeile mit Line Input können mit Input auch Werte auf einzelne Variablen gelesen werden
- Aus dem Text

  213 12 23.5 Ausfall

  kann mit

```
Dim  i as integer, j as integer , d as double , s as string
Open ...
    Input #Dateinr,  i, j, d , s
```

  der obige Inhalt auf die Variablen gelesen werden.

PROBLEM: bei fehlendem Werten in der Datei kann es zu Fehlern kommen, besser ist eine zeilenweise Verarbeitung mit Kontrolle durch das Programm !

Das Schreiben von Dateien

- Nach einem erfolgreichem Öffnen einer Datei im Schreib- oder Appendmodus kann mit Print #Dateinr, s geschrieben werden
- Die Menge der Daten wird i.d.R. durch das Programm bestimmt !

```
Dateinr = FreeFile: Rem Hole freies Handle
Dim  Zeilen(1000) as string
...  Setze einen Inhalt für die N-Zeilen
Open Dateiname For Output As Dateinr
        for i = 1 to N
            Print #Dateinr,  i ;  Zeilen(i)
        next i

 Close Dateinr  : Rem Schließen NICHT vergessen !!
```

Fehlerbehandlung bei Dateioperationen

- Alle Dateioperationen können durch fehlende Dateien, falsche Optionen , Bedienfehler oder Hardwareproblemen zu Laufzeitfehlern führen.
- Insbesondere bei Dateioperationen ist ein STABILES Fehler-management notwendig. VB stellt generell, also auch für alle anderen bereits gezeigten VB-Funktionen, folgende Konstruktion bereit :

```
On error goto  marke1
    Rem  Möglicherweise Fehlerverursachende Funktionen
    ....
markeweiter2:  Exit sub : Rem Funktionsende
Marke1:
    rem Fehlerbehandlung
    msgbox   error (err)   :   Rem  Fehleranzeige
    resume markeweiter2 :  Rem gehe wieder hoch
```

Bei Fehler wird abgebrochen und zur marke1 gesprungen

Hilfsfunktionen für Dateioperationen (Auszug)

- Neben dem Lesen und Schreiben werden Funktionen für das Dateimanagement bereit gestellt :

- Chdir (NeuerPfadname) : Rem wechselt den Pfad
- ChDrive (NeuerLaufwerksbst) : Rem wechselt das Laufwerk
- s = CurDir(LW) : Rem liefert den aktuellen Pfad

- MKDir und RMDir generieren / löschen ein Verzeichnis

- Kill(Pfad_Dateiname) - löscht die Datei OHNE RÜCKFRAGE
- Filelen(Pfad_Dateiname) liefert die Dateilänge

Hilfsfunktionen für Verzeichnisoperationen

- Sehr nützlich sind Funktionen zum Durchlaufen der Verzeichnis-informationen auf größeren Datenträgern (Festplatte etc.)
- Das Schlüsselwort für Verzeichnisinformationen ist DIR
- Dir wird beim ERSTEN Aufruf der gewünschte Pfad übergeben und liefert (falls vorhanden) bei allen weiteren Aufrufen JE EINEN VERZEICHNISEINTRAG zurück (beim 2. – N. Aufruf ist KEIN Parameter zulässig) :

```
Dim fn As String
fn = Dir(Pfadname & Dateimuster) : ‘ 1. Aufruf mit Pfad und Erw.
While fn <> ""
  Debug.Print fn    : Rem Kontrollausgabe
  Me!Ergebnis = Me!Ergebnis & fn & Chr(13) & Chr(10)
  fn = Dir    :   Rem 2.-N. Aufruf OHNE Parameter
Wend
```

Unterscheidung von Verzeichnisoperationen

- Verzeichniseinträge können zusätzliche Attribute tragen
- Unterverzeichnisse werden GENAUSO wie Dateien abgelegt und müssen daher von diesen unterschieden werden
- unter DOS/Windows erfolgt dies durch Testen der Attribute mittels vordefinierter Konstanten :

```
If (GetAttr(Pfad & fn) And vbDirectory) = vbDirectory Then
      REM ' Eintrag nur anzeigen, wenn es sich um ein Verzeichnis handelt
      Debug.Print „VERZEICHNIS:“  &   fn
 End If '
```

- Die gleiche Funktionalität kann analog mittels eines optionalen Parameters bei 1. Aufruf von DIR erreicht werden `fn = Dir(Pfadname & Dateimuster, vbDirectory ) : ‘ 1. Aufruf !!`

Typen von Verzeichniseinträgen

- Verzeichniseinträge können zusätzliche Attribute tragen
- zum Testen stehen folgende Konstanten bereit :

- **vbNormal** - (Voreinstellung) Dateien ohne Attribute **(Wert=**0)
- **vbReadOnly** - Schreibgeschützte Dateien, zusätzlich zu Dateien ohne Attributen **(Wert=**1)
- **vbHidden** - Versteckte Dateien, zusätzlich wie vorher (**Wert=**2)
- **vbSystem** - Systemdatei, zusätzlich wie vorher (**Wert=4**)
- **vbArchive - zeigt Änderungen an (-> für Backups !!)**
- **vbVolume -** Datenträgerbezeichnung (**Wert=8**)

- **vbDirectory -** Verzeichnis oder Ordner, zusätzlich zu Dateien ohne Attributen (**Wert=16**)