

Daten auf Scheiben

File- und Diskettenstrukturen unter CP/M, MSDOS und TOS

Wilfried Haaf
Frank Middel

Die Hauptaufgabe eines Disketten-Betriebssystems ist die Verwaltung und Zuteilung des verfügbaren Diskettenplatzes zu den verschiedenen Dateien. Der Anwender wird dadurch von der Buchführung über belegte und verwendete Sektoren entlastet. Er kennzeichnet seine Daten mit einem Namen, und alles andere erledigt das Betriebssystem. Hier sind die Prinzipien dargestellt, nach denen die drei meistgebräuchlichen Betriebssysteme dabei vorgehen.

Der Anwender oder der Hochsprachen-Programmierer benötigt diese Informationen im Normalfall nicht. Er kann sich darauf verlassen, daß das Disketten-Betriebssystem Daten zuverlässig abspeichert und sicher wiederfindet. Die hier dar-

gestellten Prinzipien geben aber einen guten Einblick in die Arbeitsweise des DOS (Disk Operating System), und manche Eigenheiten und Reaktionen des Betriebssystems können dadurch besser verstanden und interpretiert werden.

Dateiverwaltung bei CP/M

Der Urvater aller Betriebssysteme für Mikrocomputer zeichnet sich dadurch aus, daß der Systementwickler viele Parameter frei wählen kann. Dadurch lassen sich an einem CP/M-Rechner Disketten- und Festplatten-Laufwerke mit unterschiedlichsten Eigenschaften und Kapazitäten verwenden. Leider führte diese Freiheit auch dazu, daß es kaum zwei CP/M-Rechner mit kompatibelem Diskettenformat gibt. Bei den 5,25"-Formaten ist die Vielfalt besonders beeindruckend (siehe Chaos mit System, c't 6/85, S.120). Nur bei den

8"-Formaten existiert ein Quasi-Standard, der hier als Beispiel zur Erläuterung des Prinzips dient.

Das 8-Zoll-Standard-CP/M-Format wird in 'Single-Density' (FM) beschrieben und hat 77 Tracks mit je 26 Sektoren zu 128 Byte. Damit stellt dieses Format eine Kapazität von $77 \times 26 \times 128 \text{ Byte} = 256.256 \text{ Byte} = 250,25 \text{ KByte}$ zur Verfügung. Die Spuren 0 und 1 enthalten das Betriebssystem selbst, für die eigentliche Datenspeicherung stehen also die Tracks 2...76 zur Verfügung.

Um die Zuordnung 'Datei <-> Sektoren' zu erleichtern, werden immer 8 Sektoren zu einem 'Block' gebündelt. Block 0 belegt demnach die ersten 8 Sektoren auf Track 2, Block 1 die nächsten 8 Sektoren und so weiter. Der Rechner kann aus der Blocknummer relativ einfach die dazugehörenden Sektornummern ermitteln.

Zur Verwaltung der Diskette benützt CP/M zwei Hilfsmittel:

- Das Directory enthält für alle Dateien der Diskette die Informationen über Dateinamen und belegte Blöcke. Diese Buchführung steht in den ersten 16 Sektoren von Track 2, also in den beiden CP/M-Blöcken 0 und 1.

- Die Disk Allocation Map enthält für jeden Block der Diskette ein Bit. Eine '1' zeigt, daß dieser Block belegt ist, eine '0' gibt an, daß dieser Block frei ist. Die Map steht im RAM des Rechners, das heißt, sie muß bei einem Diskettenwechsel neu an-

gelegt werden. (Deshalb das Ctrl-C nach jedem Diskettenwechsel!)

Jeder Directory-Eintrag besteht aus 32 Byte, deren Bedeutung aus dem abgebildeten Schema ersichtlich ist. Nichtbelegte Blöcke haben immer die Nummer 0, dies ist möglich, weil im Block 0 (und im Block 1) im Normalfall das Directory enthalten ist. Ein Directory-Sektor von 128 Byte faßt genau 4 Einträge. Da beim Standard-CP/M 16 Sektoren (die Blöcke 0 und 1) für das Directory reserviert sind, kann eine CP/M-Diskette also maximal 64 Dateien verwalten.

Die Zahl der Directory-Einträge kann allerdings genau wie die Anzahl der Tracks pro Diskette, die Anzahl der reservierten Spuren, die Anzahl der Sektoren pro Track und die Blockgröße in weiten Grenzen geändert werden. Auch kann CP/M bei mehr als 256 Blöcken pro Diskette 16-Bit-Zahlen für die Block-Numerierung verwenden. Pro Eintrag sind dann nur noch 8 Verweise auf belegte Blöcke möglich. Um die Transfer-Geschwindigkeit zu erhöhen, ist es besonders bei

```
0058595A 20202020 20424148 00000003 *.XYZ BAK....*
02000000 00000000 00000000 00000000 *......*
E5444953 4B564552 31484C50 00000000 *.DISKVER1HLP....*
00000000 00000000 00000000 00000000 *......*
0058595A 20202020 20484C50 00000000 *.XYZ HLP....*
05060708 0F101112 13141516 1718191A *......*
0058595A 20202020 20484C50 01000023 *.XYZ HLP....*
1B1C1D1E 1F000000 00000000 00000000 *......*
```

5,25"-Formaten üblich, mit größeren Sektoren von 512 oder 1024 Byte zu arbeiten. Dann sind allerdings besondere Maßnahmen erforderlich, um CP/M die Diskettendaten weiter in 128-Byte-Portionen anbieten zu können (Blocking/Deblocking). Näheres zur Anpassung von CP/M an verschiedene Diskettenformate kann man etwa in c't 4/87, S. 172: 'Die Diskparameter von CP/M 2 und 3' oder in der Serie 'Einsteigen in CP/M' (c't 7/85 bis c't 1/86) nachlesen.

Große Dateien verwalten

Beim 8"-Format verwaltet jeder Directory-Eintrag maximal 16 Blöcke (also 16 KByte). Größere Dateien teilt CP/M in 16 KByte große Bereiche, sogenannte Extensions, für die jeweils ein eigener Eintrag angelegt wird. Dabei dient das ex-

nachdem auf einer neu formatierten Diskette drei Dateien angelegt wurden und anschließend wieder eine gelöscht wurde. Der erste Eintrag verwaltet die Datei XYZ.BAK. Er ist gültig und belegt 3 Sektoren. Diese Sektoren liegen im Block Nummer 02. Der zweite Dir-Eintrag ist ungültig, denn das erste Byte hat den Wert E5.

```
Index: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12
Inhalt: 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1

Index: 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25
Inhalt: 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
```

Mit der Disk Allocation Map ermittelt CP/M die freien Blöcke

Die beiden folgenden Einträge verwalten die Datei XYZ.HLP. Da diese Datei aus 80hex + 23hex = A3hex = 163dez Sektoren ($\cong 20,375$ KByte) besteht, hat CP/M 2 Extensions angelegt. Die erste Extension hat an der Stelle ex den Eintrag 00, die zweite Extension hat dort den Wert 01.

An der Tatsache, daß die zweite Extension 'nur' 23hex Sektoren verwaltet, kann man erkennen, daß dies die 'letzte Extension' dieser Datei ist. Die 163 Sektoren ($\cong 21$ Blöcke zu 1 KByte) liegen in den Blöcken 05, 06, 07, 08 und 0Fhex bis 1Fhex. Um freie Sektoren schnell zu ermitteln, bedient sich CP/M nicht der Directory-Informationen, sondern einer zweiten Datenstruktur.

Disk-Allocation-Map

Im Directory sind eigentlich alle Informationen enthalten, die für eine Dateiverwaltung benötigt werden. Wenn CP/M eine Datei sucht (etwa beim Öffnen einer Datei), werden alle dort abgelegten Dateinamen mit dem gesuchten Namen verglichen. Existiert ein gültiger Ein-

trag mit dem gesuchten Namen, läßt das Betriebssystem die 16 Blocknummern und berechnet die Reihenfolge und die Nummer der verwendeten Sektoren.

Beim Suchen nach einem freien Block, zum Beispiel bei der Neuanlage oder Erweiterung einer Datei, kann die Arbeit, allein mit dem Directory, allerdings

sehr mühselig sein. Um diesen Vorgang zu erleichtern, legt CP/M für jedes Laufwerk einen reservierten RAM-Bereich an. Dieser Bereich, die Disk Allocation Map, repräsentiert die Blöcke einer Diskette durch jeweils ein Bit. Bei einem Neustart (egal ob Kalt- oder Warmstart) liest CP/M das vollständige Directory und setzt die Bits der belegten Blöcke auf 1, alle anderen Blöcke bekommen eine 0. Im abgebildeten Beispiel wurde angenommen, daß die restlichen Sektoren des Directory, die im Hex-Dump nicht dargestellt sind, keine weiteren gültigen Einträge enthalten. Nur dann ist sichergestellt, daß die Blöcke 3, 4 und 5 frei sind, da CP/M beim Zuteilen von Blöcken immer von unten beginnt.

Aus dieser Map kann CP/M sehr schnell feststellen, ob ein Sektor frei ist oder nicht. Bei der Suche nach einem freien Block sucht CP/M die erste 0 in der Map. Der Index dieser 0 entspricht dann der Nummer des freien Blocks. Dieser Block wird nun von CP/M belegt, in den Eintrag der entsprechenden Datei eingetragen und in der Map als belegt markiert. Nach diesem Verfahren würden also bei unserer Beispieldiskette als nächstes die Blöcke 3, dann 4, danach 9 und so weiter belegt.

Da CP/M die Map im RAM und nicht etwa auf der Diskette anlegt, kann es einen freien Block sehr schnell ermitteln (ein RAM-Zugriff ist um mehrere

kz	n0	n1	n2	n3	n4	n5	n6	n7	t0	t1	t2	ex	s0	s1	rc
b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	bA	bB	bC	bD	bE	bF
kz	= Kennzeichen, ob der Eintrag gültig ist oder nicht. Bei kz=00...0Fhex ist der Eintrag gültig, bei kz=E5hex ist der Eintrag ungültig. So wird, z.B. beim Löschen einer Datei, deren kz-Byte im DIR-Eintrag auf E5hex gesetzt. Das Kennzeichen Byte ist gleichzeitig die Kennung für die "User-Nummer" mit der CP/M die Disk-Kapazität in 16, voneinander unabhängige "User" einteilen kann.														
n0...n7	= Sind die (maximal) 8 Zeichen des Dateinamens in ASCII. Werden weniger als 8 Zeichen für den Dateinamen verwendet, dann wird dieser "rechts" mit "Blanks" (20hex) aufgefüllt.														
t0...t2	= Sind die 3 Zeichen des Dateitypes (Extension), ebenfalls in ASCII und, wenn nötig, mit Blanks aufgefüllt.														
ex	= Ist der Extension-Zähler. Seine Bedeutung wird später erklärt.														
s0...s1	= Diese beiden Byte sind reserviert für systeminterne Zwecke.														
rc	= Record Count, gibt die Anzahl der Sektoren (Records) an, die von dieser Datei (genauer von dieser Extension) belegt sind.														
b0...bF	= Dies sind die Blocknummern, die von der Datei belegt werden. Die Blocknummern (jeder Block hat 8 Sektoren) können in beliebiger Reihenfolge aufgeführt sein. Die ersten 8 Sektoren liegen im Block b0, die nächsten 8 Sektoren im Block b1, usw.														

Der Aufbau eines Directory-Eintrags bei CP/M

Byte im Directory-Eintrag als Zähler. Die erste Extension hat die Nummer ex=00, die zweite den Wert ex=01 und so weiter. Zusätzlich gilt, daß eine neue Extension nur angelegt wird, wenn die alte Extension voll ist, das heißt 128 Sektoren ($\cong 16$ KByte) enthält.

Das Beispiel eines Directory-Sektors zeigt die Situation,

Größenordnungen schneller als ein Disk-Zugriff). Problematisch wird es allerdings, wenn man während des Betriebes die Diskette wechselt, denn dann steht in der Map noch die Belegung der alten Diskette (und diese stimmt natürlich nicht mit

der Belegung der neuen Diskette überein!). Aus diesem Grund muß man nach jedem Diskettenwechsel mit einem Ctrl-C einen Warmstart einleiten. Sonst erhält man die berühmte Fehlermeldung 'BDOS ERR on A: Read Only'.

Dateiverwaltung unter PC/MSDOS

Jede unter MS/PCDOS formatierte Diskette beginnt auf Spur 0, Seite 0 mit dem Boot-Sektor, der bei tatsächlich bootfähigen Disketten einen Versionshinweis mit OEM-Identifikation, die BIOS-Parametertabelle sowie die DOS-Laderoutine, den eigentlichen Booter, enthält. Der Booter wird beim Kaltstart vom ROM-BIOS geladen und anschließend aktiviert. Er holt dann seinerseits das Betriebssystem aus den 'versteckten' Dateien IO.SYS (Disk-BIOS) beziehungsweise MSDOS.SYS

(DOS-Funktionen). Diese Dateinamen können von Rechner zu Rechner verschieden sein. PCDOS verwendet zum Beispiel die Bezeichnungen IBMBIO.COM und IBMDOS.COM. Nicht bootfähige Disketten bekommen einen Hinweis auf das Fehlen des Betriebssystems.

Ähnlich wie CP/M verwendet MSDOS für die Disk-Verwaltung zwei Hilfsmittel: File Allocation Tables (FAT) und Directories. In den FATs ist vermerkt, welche Bestandteile zu einer Datei gehören. Da dies eine sehr wichtige Information zur Verwaltung des Diskettenspeicherplatzes ist, wird zur Sicherheit das Original und mindestens eine Kopie angelegt. Die erste FAT schließt sich direkt an den Boot-Sektor an. Abhängig vom Hersteller können zur obligatorischen ersten Kopie weitere FAT-Kopien kommen. PCDOS (IBM) arbeitet grundsätzlich nur mit einer Kopie.

Auf jeder Diskette befindet sich weiterhin das Root-Directory. Die Größe des Stammverzeichnisses ist während des Formatierens variabel definierbar. Sie wird durch einen Eintrag im Boot-Sektor festgelegt. Man kann jedoch die Fehlermeldung erhalten, daß keine weiteren Einträge ins Root-Directory passen, da es während des normalen Betriebes nicht mehr erweiterbar ist. MSDOS zeigt sich seit Version 2.0 durch seine Unter-Directories sehr flexibel, da es diese in der Disk-Verwaltung wie normale Dateien behandelt, die ja nicht überlaufen können.

Sektoren und Cluster

Das MSDOS-Standardformat verwendet doppelseitige Disketten mit 2 x 40 Tracks. Auf jedem Track befinden sich 9 Sektoren zu je 512 Byte. Damit hat die Diskette eine Kapazität von 2 x 40 x 9 x 512 Bytes = 360 KBytes. Mit den Bezeichnungen für die Sektoren kann es leicht zu Ver-

```
A>DEBUG                                Laden des Debuggers

-L adr lw start anzahl                lädt (L)
                                         nach Adresse <adr>
                                         vom Laufwerk <lw>
                                         beginnend mit Sektor <sect>
                                         eine <anzahl> Sektoren

-D von bis                             listet den Speicherinhalt

Beispiele:

-L100 0 1 1                           lädt 1 Sektor ab logischem Sektor 1
                                         (Side 0, Track 0, physikalischer
                                         Sektor 2) von Disk 0 (A:) in den
                                         DEBUG-Speicher ab 100h

-L100 1 5 3                           lädt 3 Sektoren vom 5. logischen Sektor
                                         (Side 0, Track 0, physikalischer
                                         Sektor 6) von Disk 1 (B:) nach 100h.
                                         Dies sind im 360-KB-Standardformat die
                                         ersten drei Directory-Sektoren

-L 0 0 0 1                             lädt den Boot-Sektor von Laufwerk A:
```

Mit dem Dienstprogramm DEBUG, das normalerweise zum Lieferumfang von MSDOS gehört, kann man sich jeden Sektor in den Speicher laden und ausgeben lassen.

baren Speichereinheiten mit 1 KByte Größe numeriert man einfach durch. Allerdings hat das erste Cluster die Nummer 2, das zweite die Nummer 3 und so weiter. Es wird mit der Nummer 2 begonnen, weil die ersten beiden FAT-Einträge eine besondere Bedeutung haben. Beim 360-K Byte-Standardformat sind die ersten 12 Sektoren reserviert: 1 Boot-Sektor, 4 (2 x 2) FAT-Sektoren und 7 Directory-Sektoren. Damit kann man sich aus der Cluster-Nummer relativ schnell die logische Sektor-Nummer berechnen:

$$\text{snr} = (\text{cnr} - 2) * 2 + 12$$

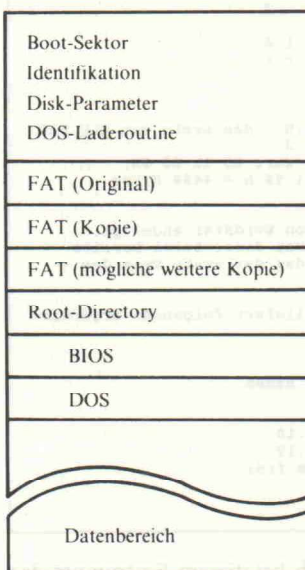
Directories

Für jede Datei legt das Betriebssystem einen 32 Byte langen Eintrag in einem reservierten Teil der Diskette an, wobei zu beachten ist, daß numerische Werte, wie bei Intel-Prozessoren üblich, mit dem niederwertigen Byte zuerst abgespeichert sind. Den Aufbau und die Auswertung eines Directory-Eintrags zeigen zwei Tabellen.

File Allocation Table (FAT)

Das erste Cluster, das einer Datei zugewiesen wird, ist im Directory enthalten. Die Information über das Nachfolge-Cluster befindet sich in der FAT. Man kann sich die FAT als Tabelle vorstellen, deren Fächer von 0 bis 4095 durchnummeriert sind. Jedes Fach entspricht einem Cluster und enthält Informationen, die

- entweder die nächste Cluster-Nummer anzeigen
- oder das Dateiende
- oder anzeigen, daß dieses Cluster frei ist,



Die Struktur einer MSDOS-Diskette. Da MSDOS herstellerspezifisch konfiguriert werden kann, ist die Anzahl der FAT-Kopien nicht festgelegt. Im allgemeinen arbeitet man aber wie unter PCDOS nur mit einer Kopie. Disketten-Versionen mit BIOS- und DOS-Dateien sind nur auf bootfähigen Disketten vorhanden.

n0	n1	n2	n3	n4	n5	n6	n7	t0	t1	t2	at	r0	r1	r2	r3
r4	r5	r6	r7	r8	r9	zl	zh	dl	dh	cl	ch	l0	l1	l2	l3

n0...n7 Der ASCII-verschlüsselte Dateiname. Namen mit weniger als acht Zeichen stehen linksbündig und werden mit Blanks (20h) aufgefüllt.
Bedeutung des ersten Zeichens:
00h dieser Eintrag wurde noch nie benutzt
05h zeigt an, daß das erste Zeichen des Dateinamens E5h (o) ist
2Eh Unterverzeichnis
E5h Datei wurde gelöscht

t0...t2 Die drei Zeichen des Dateityps. Ebenfalls linksbündig mit Blanks aufgefüllt.

at Dateiattribut
Bit 0 = "Read Only": Datei kann nur gelesen werden
Bit 1 = "Hidden File": Dateiname wird im Directory nicht aufgelistet.
Bit 2 = "System File": Datei kann nicht kopiert oder gestartet werden und erscheint nicht im Directory. Zwischen Bit 1 und Bit 2 besteht eigentlich kein Unterschied.
Bit 2 wurde nur zur Kompatibilität mit anderen Systemen eingeführt.
Bit 3 = "Volume ID": kennzeichnet Eintrag als den Namen einer Diskette
Bit 4 = kennzeichnet den Eintrag als Unterverzeichnis
Bit 5 = "Archive": zeigt BACKUP, daß diese Datei seit der letzten Datensicherung verändert wurde. Es wird beim Erstellen oder Verändern einer Datei gesetzt und nach dem Backup wieder gelöscht.
Bit 6 reserviert
Bit 7 reserviert

r0...r9 für zukünftige Erweiterungen reserviert

zl...zh Uhrzeit der letzten Dateiänderung

Bit 15
h h h h h m m m m m s s s s
h = Stunde (0...23)
m = Minute (0...59)
s = Sekunde/2

dl...dh Datum der letzten Dateiänderung

Bit 15
j j j j j j m m m m t t t t
j = Jahreszahl - 1980
m = Monat (1...12)
d = Monatstag (1...31)

cl...ch Erster für die Datei reservierter Cluster

l0...l3 Länge der Datei in Bytes. Wie bei Intel-Systemen üblich, wird zuerst das niederwertige Byte abgespeichert.

Auf einer neu formatierten Datendiskette wurden mit WordStar zwei Dateien angelegt. Das mit DEBUG ermittelte Directory hat folgenden Aufbau:

```

48 4C 50 20 20 20 20 20-42 41 4B 20 00 00 00 00 HLP BAK ....
00 00 00 00 00 00 44 5A-A2 0C 02 00 80 11 00 00 .....DZ".....
48 4C 50 20 20 20 20 20-54 58 54 20 00 00 00 00 HLP TXT ....
00 00 00 00 00 00 76 5A-A2 0C 07 00 00 69 00 00 .....vZ".....i...
E5 44 42 41 43 4B 55 50-24 24 24 20 00 00 00 00 eBACKUP$$$ ....
00 00 00 00 00 00 6A 5A-A2 0C 10 00 00 10 00 00 .....JZ".....
00 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 .vvvvvvvvvvvvvvvv
F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 vvvvvvvvvvvvvvvv

```

Beim Formatieren der Diskette wurde F6h in die Datenbytes geschrieben. Ein Directory-Eintrag ist gültig, wenn das erste Byte ein Buchstabe oder eine Ziffer ist. E5h kennzeichnet eine gelöschte Datei, 00 kennzeichnet einen Eintrag, der noch nie verwendet wurde.

Auswertung des ersten Eintrages

n0...n7 Der Dateiname ist HLP

t0...t2 Der Dateityp ist BAK

at Die Datei hat das Attribut "archive", d.h., beim nächsten Aufruf des Backup-Programmes wird diese Datei automatisch kopiert und dieses Attribut zurückgesetzt

r0...r9 Die 10 reservierten Bytes sind 0

zl...zh Die beiden Uhrzeitbytes sind 44h 5Ah. Entschlüsselt man diese, dann erhält man

```

Bit 15
0 1 0 1 1 0 1 0 0 1 0 0 0 1 0 0
h h h h m m m m m s s s s s

```

→ 11 Uhr und 18 Minuten und 8 Sekunden

dl...dh Die beiden Datumsbytes sind A2h 0Ch. Entschlüsselt man diese, dann erhält man

```

Bit 15
0 0 0 0 1 1 0 0 1 0 1 0 0 0 1 0
j j j j j j m m m m t t t t t t

```

→ 2. Mai 1986

cl...ch Das Start-Cluster ist 02 00, d.h., das erste von HLP.BAK belegte Cluster hat die Nummer 2
l0...l3 Die vier Längenbytes haben den Wert 80 11 00 00, d.h., die Datei belegt 00 00 11 80 h = 4480 Bytes

Die gelöschte Datei EDBACKUP.\$\$\$ wurde von WordStar angelegt, als die Datei HLP.TXT bearbeitet wurde. Daß diese Datei bereits gelöscht ist, kann man am E5h erkennen, das das erste Byte des Namens überschreibt.

Das DIR-Kommando für die obige Diskette liefert folgendes Ergebnis:

B>dir

Kennsatz in Laufwerk B hat keinen Namen
Verzeichnis von B:

```

HLP BAK 4480 2.05.86 11.18
HLP TXT 26880 2.05.86 11.19
2 Datei(en) 329728 bytes frei

```

- oder defekte Bereiche der Diskette markieren.

Da die Nummer des Start-Clusters im Directory steht, kann sich MSDOS damit alle notwendigen Informationen über eine Datei berechnen. Bei einer größeren Datei steht in dem 'Fach', das dem Start-Cluster entspricht, einfach die Nummer des nächsten Clusters, in dessen 'Fach' wieder die Nummer des nächsten. Das letzte Cluster hat das Datei-Endezeichen (zum Beispiel FFFh).

Die FAT bildet die Indizes für eine 'verkettete Liste'. Das Start-Cluster im Directory-Eintrag gibt an, welches Cluster

Die Struktur eines Directory-Eintrags unter MSDOS. Neben den in der DIR-Anzeige mehr oder weniger sichtbaren Informationen über Dateinamen, Attribut, Entstehungsdatum und Länge der Datei befindet sich hier ein Zeiger auf das erste für die Datei reservierte Cluster (links).

Ein Beispiel für die Auswertung eines Directory-Eintrags (rechts).

den Beginn der Datei darstellt. Gleichzeitig gibt es die Nummer des Eintrags in der FAT an, in dem die nächste Cluster-Nummer enthalten ist. In dieser Nummer findet das Betriebssystem wieder die Nummer des nächsten Clusters und so weiter. Das letzte Cluster einer Datei bekommt wie erwähnt ein Datei-Endezeichen zwischen FF8h und FFFh. Nicht belegte, das heißt freie Cluster werden mit 000 gekennzeichnet. Sucht MSDOS freien Speicherplatz, dann nimmt es das erste Cluster, dessen Eintrag in der FAT 000 hat.

In der FAT mit dem Index 0 wird die verwendete Diskettenart abgespeichert. Es handelt

sich bei diesem Eintrag um das Media-Byte. Der zweite Eintrag ist reserviert und wird als 'Füllzeichen' bezeichnet. Für 5,25-Zoll-Disketten auf 40-Spur-Laufwerken gibt es unter anderem folgende Media-Beschreibungen:

FFh zweiseitig, 8 Sektoren

FEh einseitig, 8 Sektoren

FDh zweiseitig, 9 Sektoren

FCh einseitig, 9 Sektoren

Selbst leicht beschädigte Disketten kann man verwenden, da DOS in der Lage ist, defekte Sektoren zu markieren. Man versteht darunter Bereiche, deren Formatierung nicht einwandfrei durchzuführen ist.

Wert Bedeutung

0000h	freies Cluster
FFF0h - FFF6h	reserviertes Cluster **
FFF7h	zerstörtes Cluster
FFF8h - FFFfh	letztes Cluster
xxxxh	nächstes Cluster

** nach: Ray Duncan, MSDOS für Fortgeschrittene, Microsoft Press/Vieweg, Braunschweig 1986

Bedeutung der FAT-Einträge. Bei 12-Bit-FATs entfällt jeweils das erste Byte.

Auf einer Diskette sind 3 Dateien vorhanden.

Datei A belegt die Cluster 2, 3, 4, 5 und 7
Datei B belegt die Cluster 6, 8 und 12
Datei C belegt die Cluster 9, 10, 11, 13, 14 und 17
Alle anderen Cluster sind nicht belegt.

Das erste belegte Cluster wird im Directory-Eintrag vermerkt.

Start-Cluster Datei A: 2
Start-Cluster Datei B: 6
Start-Cluster Datei C: 9

Die FAT selbst hat dann folgenden Aufbau:

Index	Inhalt	Bedeutung
0	FFD	Formatkennzeichen
1	FFF	"Füllzeichen"
2	003	Start-Cluster: nächstes Cluster 003 [A]
3	004	nächstes Cluster 004 [A]
4	005	nächstes Cluster 005 [A]
5	007	nächstes Cluster 007 [A]
6	008	Start-Cluster: nächstes Cluster 008 [B]
7	FFF	letztes Cluster von Datei [A]
8	00C	nächstes Cluster 00Ch (12) [B]
9	00A	Start-Cluster: nächstes Cluster 00Ah (10) [C]
10	00B	nächstes Cluster 00Bh (11) [C]
11	00D	nächstes Cluster 00Dh (13) [C]
12	FFF	letztes Cluster von Datei [B]
13	00E	nächstes Cluster 00Eh (14) [C]
14	011	nächstes Cluster 011h (17) [C]
15	000	frei
16	000	frei
17	FFF	letztes Cluster von Datei [C]
18	000	frei
19	000	frei
20	000	frei

Diese Sektoren, beziehungsweise die dazugehörigen Cluster, kennzeichnet das Formatierprogramm in der FAT mit dem Wert FF7h.

DOS kennt zwei Größen für FAT-Einträge: Die Standardgröße beträgt 12 Bit. Seit DOS 3.0 gibt es für Festplatten 16 Bit breite Cluster-Nummern, die automatisch bei Kapazitäten oberhalb 10 MByte verwendet werden. Da der Umgang mit 12-Bit-FAT etwas kniffliger, das Prinzip aber gleich ist, betrachten wir im folgenden nur

noch 12-Bit-FATs, die in erster Näherung 4096 KBytes (2^{12} Cluster) verwalten können. In Wirklichkeit scheiden jedoch einige Cluster-Nummern aus, weil sie für Verwaltungszwecke reserviert sind (siehe Tabelle), so daß die Grenze für den verwalteten Speicherbereich etwas tiefer liegt.

Bei größeren Speicherkapazitäten (Festplatten) verwendet DOS nicht nur breitere FAT-Einträge, sondern auch größere Cluster, das heißt, es werden mehr als 2 Sektoren zu einem

Die FAT wurde mit DEBUG ermittelt.

```
FD FF FF 03 40 00 05 60-00 FF 8F 00 09 A0 00 0B
C0 00 0D E0 00 0F 40 01-00 00 00 00 00 00 15 60
01 17 80 01 19 A0 01 1B-C0 01 1D E0 01 1F 00 02
21 20 02 23 40 02 25 F0-FF 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00-00 00 00 00 00 00 00
```

Schreibt man jeweils drei Bytes in der richtigen Reihenfolge, dann kann man die einzelnen Einträge leicht entschlüsseln:

24-Bit-Wort	Index	Eintrag	Bedeutung
FF FF FD =>	0	FFD	Formatkennzeichen
	1	FFF	Füllzeichen
00 40 03 =>	2	003	Start-Cluster von HLP.BAK
	3	004	
00 60 05 =>	4	005	
	5	006	
00 8F FF =>	6	FFF	letztes Cluster von HLP.BAK
	7	008	Start-Cluster von HLP
00 A0 09 =>	8	009	
	9	00A	
00 C0 0B =>	A	00B	
	B	00C	
00 E0 0D =>	C	00D	
	D	00E	
01 40 0F =>	E	00F	
	F	014	nächstes Cluster ist 14
00 00 00 =>	10	000	frei
	11	000	frei
00 00 00 =>	12	000	frei
	13	000	frei
01 60 15 =>	14	015	
	15	016	
01 80 17 =>	16	017	
	17	018	
01 A0 19 =>	18	019	
	19	01A	
01 C0 1B =>	1A	01B	
	1B	01C	
01 E0 1D =>	1C	01D	
	1D	01E	
02 00 1F =>	1E	01F	
	1F	020	
02 20 21 =>	20	021	
	21	022	
02 40 23 =>	22	023	
	23	024	
FF F0 25 =>	24	025	
	25	FFF	letztes Cluster von HLP.TXT
00 00 00 =>	26	000	
	27	000	
00 00 00 =>	28	000	
	29	000	

DEBUG kann bei der FAT-Analyse hilfreich sein.

Die Struktur der 'verketteten Liste' File Allocation Table

Cluster gebündelt. Die Zuordnung Sektoren/Cluster darf immer nur eine Zweierpotenz sein, mögliche Werte sind damit 1, 2, 4, 8 und so weiter. Festplatten im IBM XT verwaltet DOS über Cluster mit 8 Sektoren (4 KByte), im AT sind es 4 Sektoren. Die Speicherbereiche für FAT und Directory auf der Platte vergrößern sich dann natürlich ebenfalls. Die XT-Festplatte benutzt unter PC DOS für die beiden FATs die Sektoren 1 bis 8 und 9 bis 16 sowie die Sektoren 17 bis 48 für das Directory.

Die FAT wird nach einem (für den Menschen) relativ komplizierten Verfahren abgespeichert. In 12-Bit-FATs werden immer zwei Einträge zu einem 24-Bit-Wort (3 Bytes) zusammengefaßt. Dabei steht der ungerade Eintrag links und der gerade rechts. Diese drei Bytes sind, wie bei Intel-Prozessoren üblich, mit dem niederwertigen Byte zuerst abgelegt.

Betrachten wir dazu zwei FAT-Einträge:

Index	Inhalt
10	00B
11	00D

Der Eintrag mit dem geraden Index und der Eintrag mit dem ungeraden Index ergeben ein 24-Bit-Wort:

00D 00B -> 00 D0 0B

Diese drei Bytes kommen nun mit dem niederwertigen Byte zuerst in den Speicher bezie-

hungsweise in die FAT auf der Scheibe:

0B D0 00

Das Entschlüsseln eines FAT-Eintrags verdeutlicht ein Beispiel.

Zerstückelte Dateien

Hiermit sind nicht Dateien gemeint, die durch irgendeinen Hard- oder Softwarefehler 'in Stücke' gegangen sind, sondern solche, die sich nicht in zusammenhängenden Speicherblöcken auf der Diskette befinden. Der Grund für das Auftreten dieser Konstellation liegt im Betriebssystem; genauer: in dem Teil des Betriebssystems, das in der FAT nach freien Speicherbereichen sucht.

Bei der Neuanlage oder Erweiterung einer Datei sucht DOS so lange in der FAT, bis es einen Eintrag mit dem Wert 000 findet. Dessen Index ist dann das erste freie Cluster. Zerstückelte Dateien entstehen vor allem in zwei Fällen:

- Wenn man auf einer relativ vollen Diskette viele kleine Dateien löscht, gibt DOS die nicht mehr benötigten Cluster frei. Kopiert man jetzt eine größere Datei auf diese Diskette, dann füllt DOS zuerst die Lücken. Dabei kann es

vorkommen, daß eine Datei zum Beispiel die Cluster 7, 49, 118 und 288 belegt. Wenn diese Datei nun bearbeitet wird, sind relativ viele Disk-Positionierungen notwendig: die Zugriffszeit sinkt.

- Zerstückelte Dateien entstehen auch, wenn man auf einer relativ vollen Diskette eine größere Datei editiert und häufig ändert, Text einfügt oder löscht.

Wenn man mit einem normalen Kopierprogramm wie COPY eine Diskette auf eine andere, leere kopiert, dann werden alle Dateien auf der neuen Diskette in aufeinanderfolgenden Bereichen abgespeichert. Das Kopierprogramm überträgt die einzelnen Dateien nacheinander, wobei immer erst eine Datei vollständig abgearbeitet wird und erst dann die nächste folgt. Damit belegen alle Daten einer Datei automatisch einen aufeinanderfolgenden Speicherbereich auf der neuen Diskette. Die alte Diskette kann man nun sehr gut zur Datensicherung beziehungsweise als Backup-Disk verwenden. Das Kopieren darf allerdings nicht 'physikalisch' erfolgen (zum Beispiel mit DISKCOPY), denn diese Programme kopieren ja Sektor für Sektor, übernehmen also auch die Stückelung.

Dateiverwaltung unter TOS

Das dritte Dateiverwaltungssystem ist das TOS (Tramiel Operating System) des Atari ST. Es wurde wie CP/M von der Firma Digital Research entwickelt. Wer nun denkt, hinter TOS stecke womöglich ein ganz neues System der Dateiverwaltung, der irrt gewaltig. Denn was sich in Wirklichkeit hinter der Bezeichnung TOS verbirgt, läßt sich einfach beschreiben: ein MSDOS-Clone.

Zur Ehrenrettung von TOS sei jedoch gesagt, daß das gerade in dieser Zeit so strapazierte Wort Clone vielleicht insofern ein wenig unzutreffend ist, als es sich bei TOS nicht um ein reinrassiges Duplikat des MSDOS-Formats handelt, das jedoch über weite Strecken kompatibel ist.

So macht es unter Verwendung geeigneter Laufwerke und Umschaltung der Steprate von drei auf sechs Millisekunden keine

Schwierigkeiten, unter TOS MSDOS-Disketten zu lesen und zu schreiben.

Die auffälligsten Parallelen zwischen TOS- und MSDOS-Datensystemen sind:

- Verwendung eines hierarchischen Inhaltsverzeichnis
- Verwaltung von Datums- und Zeitinformation zu jedem Directory-Eintrag
- Speicherung der Verkettungsinformationen des Datenbereichs der Diskette in einer FAT (File Allocation Table)

Standardgemäß wird eine TOS-Diskette mit 80 Tracks

Logischer Sektor	Inhalt
00	Boot-Sektor
01 ... 05	FAT Nr. 1
06 ... 10	FAT Nr. 2
11 ... 17	Directory
18 ... 719 (1439)	Datenbereich

Inhalt/Funktion:	Offset:
BRA.S <Adr>	Sprung zum Boot-Programm \$00
Füller	Reserviert \$02
Serien-Nummer	24 Bit lange Seriennummer, die beim Formatieren erzeugt wird \$08
BPS	Anzahl der Bytes pro Sektor \$0B
SPC	Anzahl der Sektoren pro Cluster \$0D
RES	Anzahl der reservierten Sektoren \$0E
NFATS	Anzahl der File Allocation Tables \$10
NDIRS	Anzahl der Directory-Einträge \$11
NSCTS	Anzahl der Sektoren \$13
MEDIA	Das sogenannte Media-Descriptor-Byte. Wird vom BIOS des Atari ST nicht benutzt \$15
SPF	Anzahl der Sektoren pro FAT \$16
SPT	Anzahl der Sektoren pro Track \$18
NSIDES	Anzahl der Seiten des Speichermediums \$1A
NHID	Anzahl der "versteckten" Sektoren \$1C
BOOTCODE	Ab hier steht bei einem bootfähigen Boot-Sektor die Laderoutine \$1E

Auch der Boot-Sektor unter TOS ähnelt MSDOS. Selbstverständlich wird Maschinencode für den 68000er eingesetzt.

ein- oder zweiseitig formatiert. Dabei besitzt jede Spur 9 Sektoren mit jeweils 512 Byte Speicherplatz. Daraus ergibt sich eine theoretische Speicherkapazität von 368 640 Bytes bei einseitigen und 737 280 Bytes bei doppelseitigen Disketten.

Außerdem ermöglicht TOS noch das Bearbeiten von Disketten mit einmal oder zweimal 40 Spuren. Da sich bei diesen Formaten natürlich dementsprechend weniger freier Speicherplatz ergibt, benutzt man sie in der Regel nicht. Von den 720 beziehungsweise 1440 Sektoren benötigt das Dateisystem insgesamt 18 Sektoren, wobei die Belegung wieder sehr der von MSDOS ähnelt.

Wie aus der Abbildung ersichtlich, belegt der Boot-Sektor den ersten Sektor auf jeder Diskette. Unter TOS spielt die eigentliche Aufgabe dieses Sektors, nämlich das Einladen des Betriebssystems von Diskette, keine große Rolle mehr, da sich das komplette Betriebssystem im ROM befindet. Anders verhält sich dies bei Rechnern aus der ST-Serie, die nicht über das Betriebssystem im ROM verfügen. Hier ist natürlich wieder der

Boot-Sektor gefragt, da dieser das Lade-Programm enthält. Es lädt das eigentliche Betriebssystem aus der Datei TOS.IMG.

Der Boot-Sektor enthält außer dem Loader noch weitere, für das System sehr wichtige Informationen. Sie sind für die Erkennung beziehungsweise für die korrekte Verwaltung der Platte erforderlich.

Die im Boot-Sektor enthaltenen Daten bilden den sogenannten BPB (BIOS-Parameter-Block), der unter TOS eine gewisse Kompatibilität zu MSDOS aufweist. Das bedeutet natürlich auch, daß alle 16-Bit-Zahlenwerte im Intel-8086-Format (Bytefolge: Low/High) und nicht, wie eigentlich zu erwarten, im Motorola-Format (Bytefolge: High/Low) abgelegt sind. Den Beitrag 'Damit die Scheibe spurt' in diesem Heft legen wir jedem ans Herz, der eigene Versuche mit Dateiübertragungen zwischen PC und ST vorhat.

Nach Sektor 0 folgen die zwei FATs, wobei jede insgesamt fünf Sektoren lang ist. Auch unter TOS ist die zweite FAT aus Datensicherheitsgründen eine Kopie der ersten. Die Funktion der FAT unterscheidet sich nicht von der unter MSDOS.

Die letzte und größte Einheit einer jeden Diskette ist schließlich der Datenbereich, der ab Sektor 18 beginnt und je nach Diskettenformat mit Sektor 719 oder 1439 endet.

Dateieinträge im Directory unterscheiden sich nicht von denen bei MSDOS. Auch hier ist darauf zu achten, daß alle 16-Bit-Zahlenwerte aus Gründen der Kompatibilität zu MSDOS im Intel-Format gespeichert sind.

(be/mw)

ct