RECHENZENTRUM TH MÜNCHEN
ARBEITSGRUPPE BETRIEBSSYSTEME

INTERNSCHRIFT Nr. 7

THEMA:

Das Daten-Management des IBM System/360 Operating System

VERFASSER:

Schwinn

DATUM:

17.7.1969

FORM DER ABFASSUNG

- ENTWURF
- X AUSARBEITUNG
- ENDFORM

SACHLICHE VERBINDLICHKEIT

- X ALLGEMEINE INFORMATION DISKUSSIONSGRUNDLAGE
- ERARBEITETER VORSCHLAG
- VERBINDLICHE MITTEILUNG
- VERALTET

ÄNDERUNGSZUSTAND

BEZUG AUF BISHERIGE INTERNSCHRIFTEN

Vorkenntnisse aus:

Erweiterung von:

Ersatz für:

BEZUG AUF KÜNFTIGE INTERNSCHRIFTEN

Vorkenntnisse zu:

Erweiterung in:

Ersetzt durch:

ANDERWEITIGE LITERATUR

siehe Literaturverzeichnis dieser Internschrift

Das Daten-Management des IBM System/360 Operating System

Inhaltsverzeichnis

1. Einleitung

1.1 Zweck der Internschrift ist es

a) einen Überblick über Aufbau und Funktionen des Daten-Management des "IBM System/360 Operating System" zu geben;

b) einen Wegweiser für das Studium der entsprechenden IBM Systemliteratur zu geben. Das hat zur Folge, daß anstelle detaillierter Angaben vielfach Literaturverweise stehen.

c) auf einige fragwürdige Konstruktionen hinzuweisen.

1.2 Aufgaben des Daten-Management im Operating System (OS)

Das Daten-Management bietet Möglichkeiten, um

- Datenmengen zu Informationseinheiten (Dateien) zusammenzufassen und sie durch einen Namen zu identifizieren;
- die Eigenschaften und die Struktur von Datenmengen zu charakterisieren;
- Datenmengen zu erstellen und kurzfristig oder langfristig aufzubewahren;
- Datenmengen vor unerlaubter Benutzung zu schützen;
- vorhandene Datenmengen zu ändern durch löschen und/oder hinzufügen neuer Teile (Sätze);
- Speicherplatz zu reservieren und Datenmengen dort abzulegen;
- Dateien und/oder Sätze von Dateien wieder aufzufinden.

Das Daten-Management verwaltet und führt den Transfer von Datenmengen zwischen verschiedenen Speichermedien durch.

./. 2

1.3 Literaturnachweis

Wichtigste IBM Systemliteratur ist: [4], Section II, [6], DD Statement in Section I, Section II.

2. Dateien

Eine Datei (data set) besteht aus einem Beschreibungsteil und einem Datenteil. Der Beschreibungsteil besteht aus endlich vielen Dateikennsätzen. Der Datenteil besteht aus einer Folge von endlich vielen (Daten-)Sätzen. Ein Satz (record) ist die kleinste Einheit, die vom Daten-Management adressiert und verarbeitet werden kann ([2] S. 21, 50, 54, [4] S. 58).

2.1 Dateikennsätze ([4] S. 136 - 145, [7] S. 115 - 151)

Struktur und Anzahl der Dateikennsätze hängt vom Speichermedium und der Speicherungsform (vgl. 7) ab.

Bei Dateien, die auf Speichern mit direkter Zugriffsmöglichkeit (Plattenspeicher, Magnetstreifenspeicher, Trommelspeicher) stehen, gibt es zu jeder Datei mindestens einen Dateikennsatz (= data set control block = DSCB), dessen physikalischer Speicherplatz i.a. nicht mit dem Speicherplatz des Datenteils zusammenhängt.

Bei Dateien, die auf einem Magnetband abgelegt sind, bildet der Datenteil einen zusammenhängenden Bereich und wird durch Dateikennsätze (data set header / trailer labels) begrenzt.

2.1.1 Die Struktur eines DSCB ist [7] S. 115 - 139 zu entnehmen. Er enthält im wesentlichen folgende Information

- Dateiname (bis zu 44 byte lang),
- Datenträgernummer (volume serial number),
- Erstellungsdatum, Verfallsdatum,

./. 3

- Satzformat (vgl. 2.2),
- Speicherungsform (vgl. 7),
- Schlüssellänge und -position,
- detaillierte Speicherangaben,
- Angaben, ob weitere DSCB's zur Beschreibung der Datei folgen.

2.1.2 Anzahl und Struktur der Dateikennsätze auf Magnetbändern ist [7] S. 143 - 151 zu entnehmen. Sie enthalten ebenfalls die zur Beschreibung des Datenteils notwendige Information.

2.2 Satzformate

Stellt man zum Zwecke der Ausgabe mehrere Sätze zusammen, so erhält man Blöcke. Ein Block besteht aus einem oder mehreren Sätzen ([2] S. 49, [4] S. 58).

Die Sätze einer Datei können folgende Formate haben:

2.2.1 Sätze fester Länge (Fixed-Length Records)

Jeder Satz der Datei hat konstante Länge. Satzlänge und Blockungsfaktor werden im Dateikennsatz spezifiziert ([4] S. 58 - 59).

2.2.2 Sätze variabler Länge (Variable-Length Records)

Jeder Satz und jeder Block enthält ein 2 Byte langes Längenfeld ([4] S. 59).

2.2.3 Sätze undefinierter Länge (Undefined-Length Records)

Jeder Satz wird als Block angesehen. Eine Längenangabe ist nicht vorhanden ([4] S. 60).

./. 4

3. Speicherungsmedien

Als wichtigste Speichermedien für die Ablage von Dateien werden Magnetbänder und Plattenspeicher verwendet. Magnetbänder können und Plattenspeicher müssen standardisierte Datenträgerkennsätze ([4] S. 133 - 136) enthalten. Datenträgerkennsätze stehen am physikalischen Anfang eines Datenträgers und dienen zu dessen Identifizierung. Der erste Datenträgerkennsatz enthält im wesentlichen folgende Information:

- VOL1: dient der Kennsatzidentifizierung;
- Datenträgernummer: dient der Identifizierung des Datenträgers;
- Besitzername:
- Verweis auf Inhaltsverzeichnis des Datenträgers: wird bei Magnetbändern nicht verwendet.

Es können bis zu 7 weitere Datenträgerkennsätze verwendet werden, deren Format weitgehend vom Benutzer bestimmt wird.

3.1 Magnetbandorganisation

Dateien auf Magnetbändern können auf folgende Weise abgelegt werden ([4] S. 142 - 143, [11] S. 158 - 171, 249 - 254):

a) Eine einzige Datei auf einer Magnetbandrolle.

b) Eine Datei auf mehreren Magnetbandrollen.

c) Mehrere Dateien auf einer Magnetbandrolle.

d) Mehrere Dateien auf mehreren Magnetbandrollen.

Es sei lediglich der Fall a) kurz skizziert:

./. 5

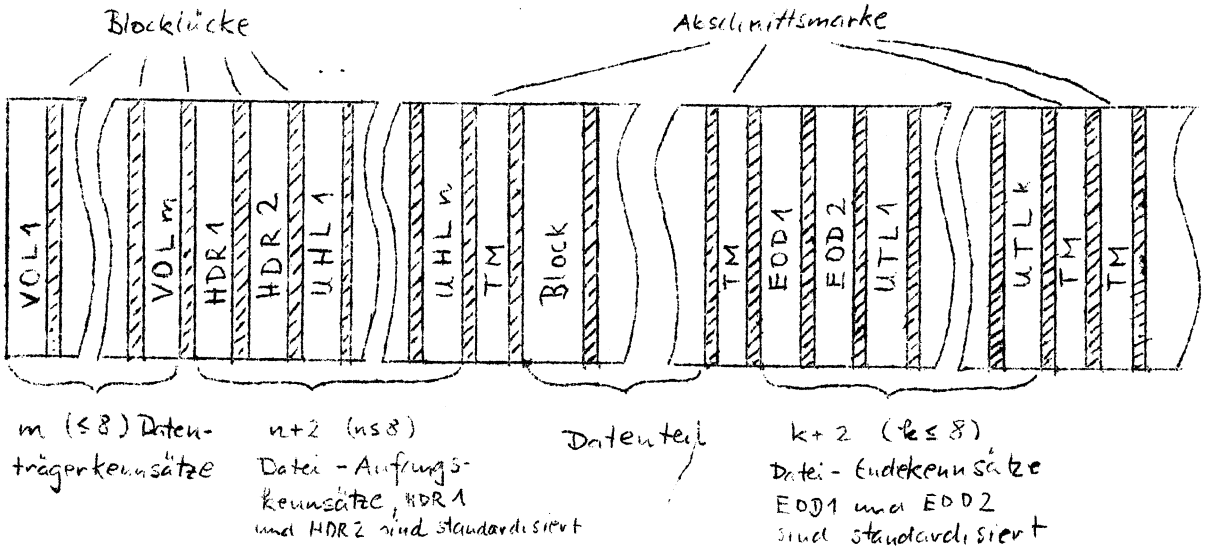


Abb. 1: Eine Datei auf einer Magnetbandrolle

3.2 Plattenspeicherorganisation

3.2.1 Gesamtorganisation ([4] S. 56, 144, [7] S. 267, [11] 259 - 260)

Jeder vom System benutzte Plattenspeicher enthält eine Gruppe von Datenträgerkennsätzen, deren erster einen Verweis auf das Datenträgerinhaltsverzeichnis (Volume Table of Contents = VTOC) enthält. Das Datenträgerinhaltsverzeichnis ist eine Datei und enthält alle Dateikennsätze der auf diesem Plattenspeicher abgelegten Dateien. Insbesondere ist der erste Satz der Dateikennsatz für das Inhaltsverzeichnis selbst. Die folgenden Sätze enthalten Information über den verfügbaren Freispeicher. Dahinter stehen die Dateikennsätze der gespeicherten Dateien.

./. 6

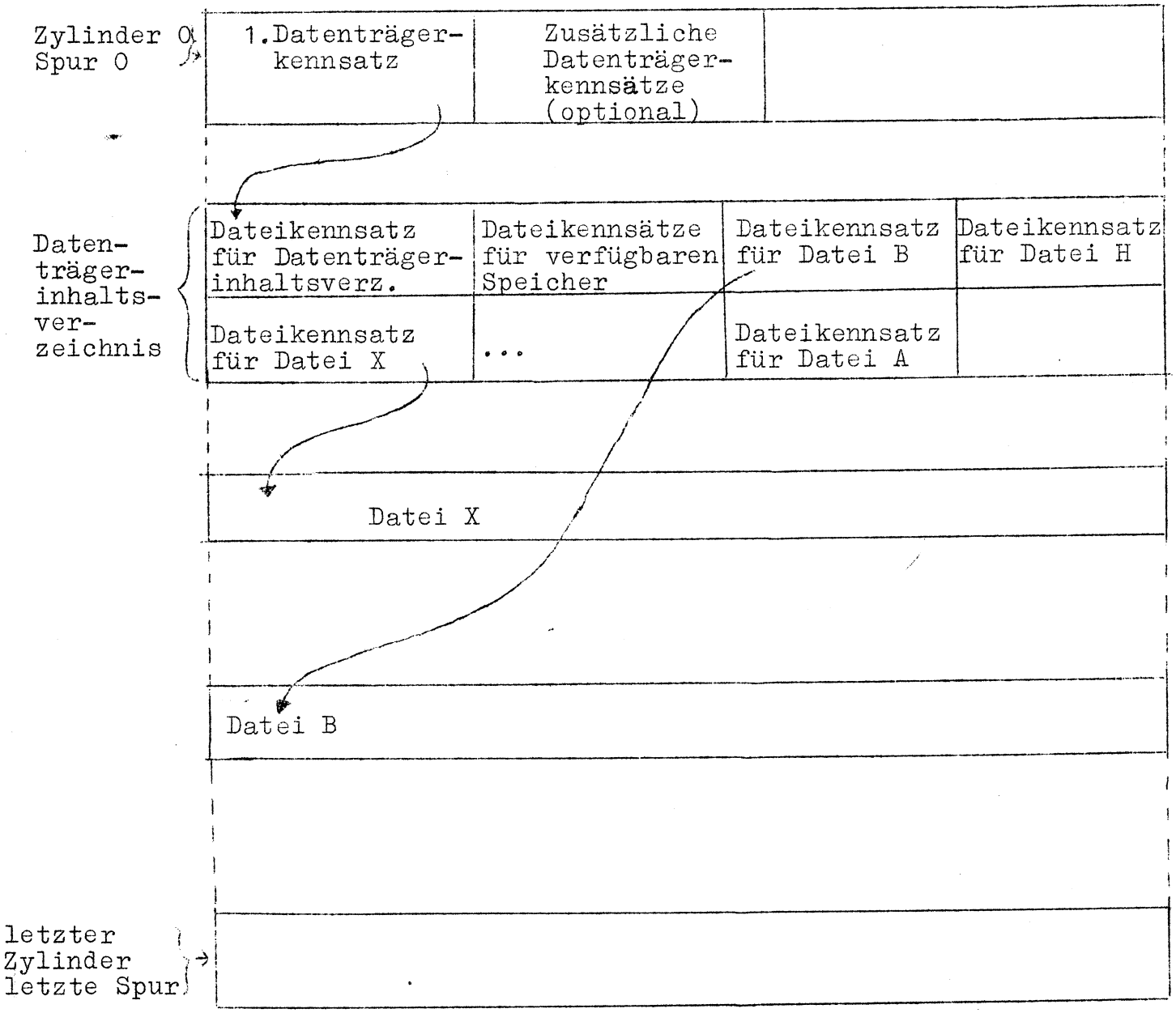


Abb. 2: Schematische Darstellung des Inhalts eines Plattenspeichers

./. 7

3.2.2 Datenorganisation auf einer Spur ([11] S. 175 - 179, [4] S. 60 - 62)

In nachstehender Abbildung ist der Aufbau einer Spur schematisch dargestellt. Hinter dem Indexpunkt, der dazu dient, den physikalischen Anfang der Spur zu kennzeichnen, folgt, getrennt durch eine Blocklücke die Spuradresse (home address). In dieser Adresse sind, außer Informationen, die Prüfzwecken dienen, Zylindernummer und Schreib/Lese-Kopfnummer der Spur enthalten. Dahinter folgt der Spurbeschreibungsbereich (Track Descriptor Record). Der Spurbeschreibungsbereich enthält Informationen darüber, wieviel Satzbereiche auf der Spur stehen und wieviel Freispeicher auf dieser Spur noch vorhanden ist. Danach folgt eine Adreßmarke. Daraufhin folgen Satzbereiche, die durch Adreßmarken und die entsprechenden Blocklücken voneinander getrennt sind.

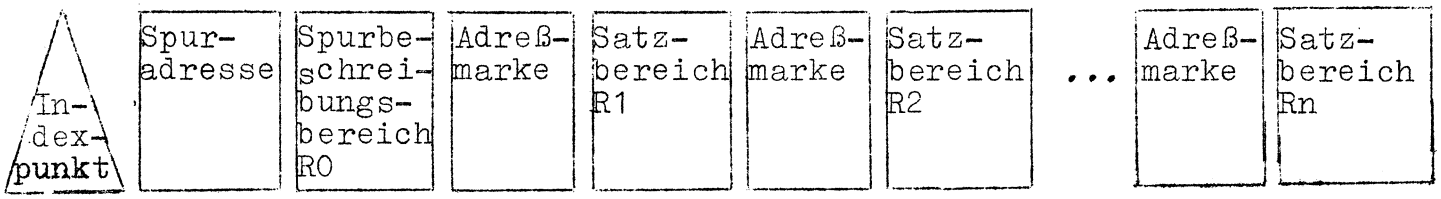
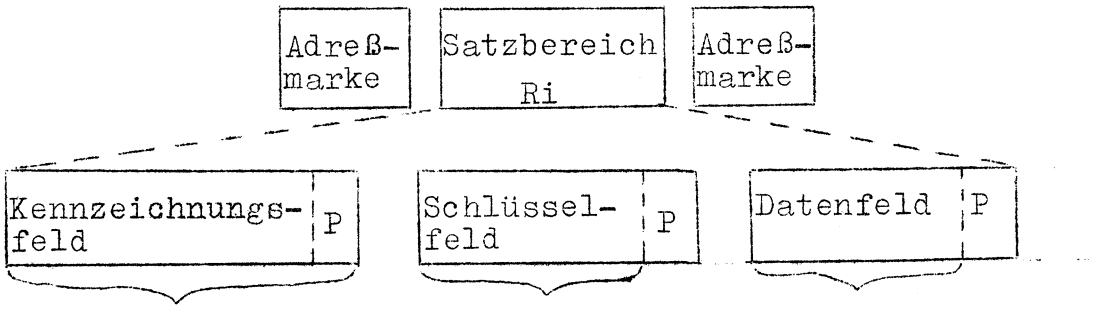


Abb. 3: Schematische Darstellung des Aufbaus einer Spur

Jeder Satzbereich zerfällt in Kennzeichnungsfeld, Schlüsselfeld (optional) und Datenfeld. Entweder hat jeder Satzbereich einer Spur ein Schlüsselfeld oder keiner hat ein Schlüsselfeld. Außer Prüfdaten enthält das Kennzeichnungsfeld, Zylinder- und Schreib/Lese-Kopfnummer, Satzbereichnummer, Schlüsselfeldlänge und Datenfeldlänge.

Der Spurbeschreibungsbereich besteht aus Kennzeichnungsfeld und Datenfeld.

./. 8



Anmerkung: P bedeutet hier ein 2 byte langes Prüfdatenfeld

Abb. 4: Aufbau eines Satzbereichs

Das Datenfeld besteht aus einem (ungeblocktes Format) oder mehreren (geblocktes Format) logischen Sätzen. Das Schlüsselfeld dient der Identifizierung des Datenfeldes. Der Inhalt des Schlüsselfeldes stimmt bei logisch fortlaufender Speicherungsform (vgl. 7.2) mit dem Schlüssel des letzten Satzes des dazugehörenden Datenfeldes überein. Auf diese Weise wird das Suchen eines Satzes aufgrund eines vorgegebenen Schlüssels wesentlich erleichtert.

4. Katalogisierungssystem ([4] S. 130 - 132, [8] S. 7 - 27)

Das Katalogisierungssystem dient dazu, Dateien langfristig aufzubewahren und sie durch Namensangabe verfügbar zu machen.

4.1 Dateinamen

Als Dateinamen können einfache Namen oder qualifizierte Namen verwendet werden. Ein einfacher Name besteht aus maximal acht alphanumerischen Zeichen, beginnend mit einem Buchstaben.

./. 9

Ein qualifizierter Name besteht aus bis zu 22 einfachen Namen, die durch "." voneinander getrennt sind ([4] S. 55). Beispiele: MAIER1, AUG.TAG10, A.B.C.D.

4.2 Aufbau des Katalogs

Der Katalog ist eine Datei, die auf einem oder mehreren Speichern mit direkter Zugriffsmöglichkeit abgelegt ist. Entsprechend der Namensbildung ist der Katalog in verschiedenen Ebenen organisiert. Die höchste Ebene ist auf der Systemresidenz (System Residence Volume) gespeichert und enthält den ersten einfachen Namen einer jeden katalogisierten Datei. Das Datenträgerinhaltsverzeichnis der Systemresidenz enthält einen Dateikennsatz, der den Teil des Katalogs beschreibt, der auf der Systemresidenz steht. Die einem einfachen Namen zugeordnete Menge einfacher Namen der nächstniedrigeren Stufe steht entweder ganz auf der Systemresidenz oder ganz auf einem weiteren Speicher mit direkter Zugriffsmöglichkeit. Der Vorteil den man hat, wenn nicht der ganze Katalog auf der Systemresidenz steht, besteht darin, daß auf der Systemresidenz Speicherplatz eingespart wird.

4.3 Funktionen des Katalogisierungssystems ([8] S. 8 - 18)

Dem Benutzer stehen Makros und Hilfsprogramme zur Verfügung, mit denen sich u.a. folgende Funktionen ausführen lassen:

- Erzeugung einer Struktur qualifizierter Namen im Katalog;
- Löschen einer Struktur qualifizierter Namen;
- Katalogisieren einer Datei;
- Löschen einer katalogisierten Datei;
- Ersetzen des Namens einer katalogisierten Datei;
- Löschen des Namens einer katalogisierten Datei;
- Lesen der Dateikennsätze einer Datei.

./. 10

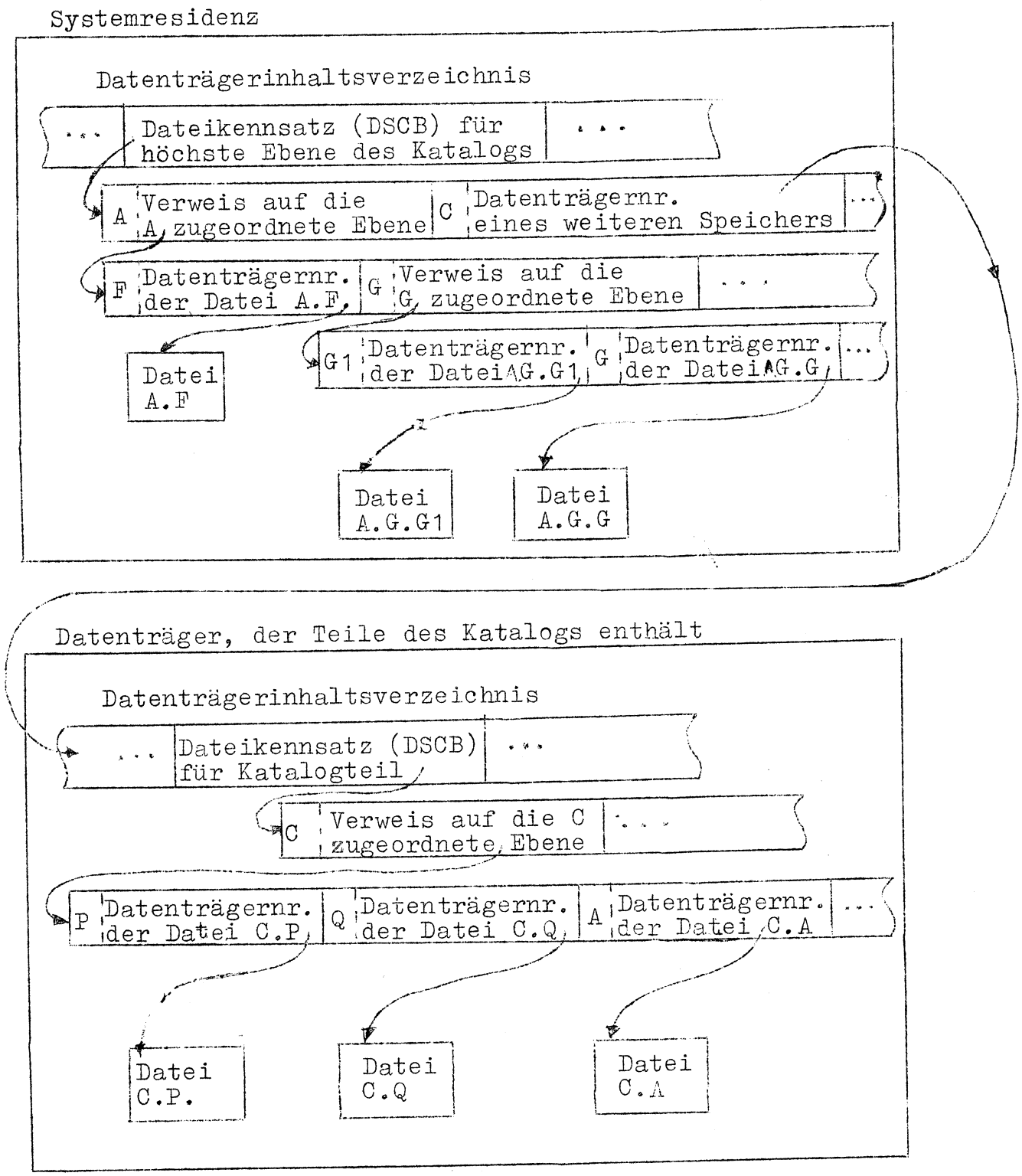


Abb. 5: Katalogstruktur mit zwei Datenträgern

5. Wo und wann werden die Eigenschaften einer Datei und Verarbeitungswünsche spezifiziert ([2] S. 21 - 23, [4] S. 63 - 70, [6] S. 24 - 78)

Bevor eine Datei verarbeitet oder erstellt werden kann, muß dem OS die dazu notwendige Information zur Verfügung gestellt werden.
Dazu gehören Angaben wie:

- Charakteristika der Datei
  Satzformat, Blockungsfaktor, Blocklänge, Speicherungsform, etc.
- Speicherangaben
  genaue Speicherangaben, wo eine vorhandene Datei abgelegt ist oder eine zu erstellende Datei abgelegt werden soll.
- Verarbeitungswünsche
  seriell, direkt, Pufferungstechniken, etc.
- Sonderwünsche
  eigene Fehlerroutinen, eigene Dateikennsatzverarbeitung, etc.

5.1 Während der Verarbeitung wird die gesamte beschreibende Information dem System im Datensteuerblock (Data Control Block = DCB, [7] S. 21 - 78) zur Verfügung gestellt.

Informationsquellen für den Datensteuerblock sind der DCB-Makro-Befehl, die Dateidefinitions-Anweisung und evtl. bestehende Dateikennsätze. Für jede Datei muß auf der Ebene der Assemblersprache ein DCB-Makro-Befehl gegeben werden. Bei Verwendung höherer Programmiersprachen erzeugt der Übersetzer einen Datensteuerblock. Für jede Datei muß eine DD-Anweisung gegeben werden.

./. 12

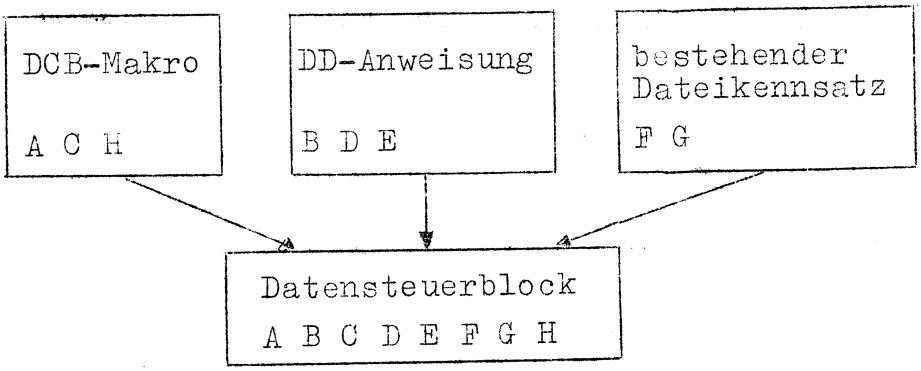


Abb. 6: Informationsquellen für den Datensteuerblock

5.2 Der Datenfluß aus den einzelnen Informationsquellen in den Datensteuerblock findet während der Ausführung des OPEN-Makros ([4] S. 75 - 77) statt. Die Hauptfunktionen der OPEN-Routine sind:

- Verknüpfung der einzelnen Informationsquellen mit dem Datensteuerblock;
- Vervollständigung des Datensteuerblocks;
- Laden der für die Verarbeitung notwendigen Zugriffsroutinen;
- Eröffnung der Datei durch Lesen oder Schreiben von Dateikennsätzen;
- Konstruktion der notwendigen Systemsteuerblöcke.

Abbildung 7 illustriert das zeitliche Auffüllen des Datensteuerblocks. Der DCB-Makro im Assemblerprogramm reserviert Platz für den Datensteuerblock und spezifiziert gewisse Felder (1). Die Information der DD-Anweisung wird vom System in einem Job-File-Steuerblock abgelegt (2). Der

./. 13

OPEN-Makro hat zur Folge, daß der Job-File-Steuerblock durch einen eventuell bestehenden Dateikennsatz modifiziert wird und danach in den Datensteuerblock übertragen wird (3,4). Durch die DCB-Ausgangs-Routine kann der Datensteuerblock vor Ausführungsende des OPEN-Makro modifiziert werden und modifizierte Information in den Job-File-Steuerblock übertragen werden (5,6). Als letztes kann ein neuer Dateikennsatz geschrieben werden (7). Für Parameter, die mehrfach spezifiziert werden, gilt:

Parameter im DCB-Makro hat Vorrang vor Parameter in der DD-Anweisung hat Vorrang vor Parameter im Dateikennsatz.

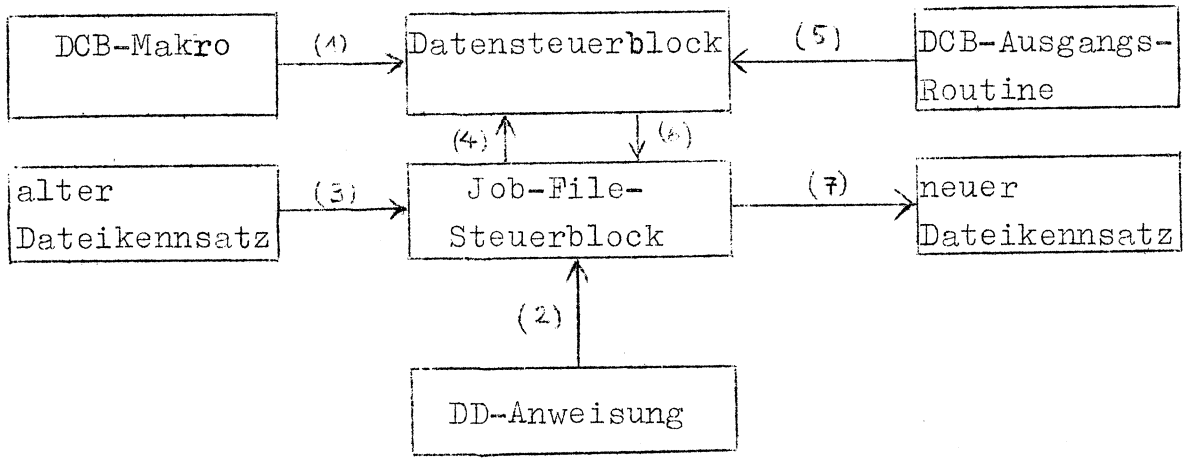


Abb. 7: Datenfluß in und vom Datensteuerblock

Die Beendigung einer Dateiverarbeitung wird durch einen CLOSE-Makro ([4] S. 77 - 78) initiiert. Diese Makro hat zur Folge, daß Dateikennsätze geschrieben werden, Enderoutinen durchgeführt werden und die logische Verbindung zwischen Datei und Datensteuerblock gelöst wird.

./. 14

5.3 Beschreibung einiger Parameter in der DD-Anweisung ([2] S. 21 - 22, [6] S. 24 - 45).

Die logische Verbindung zwischen Datensteuerblock und DD-Anweisung wird dadurch hergestellt, daß im DCB-Makro der Name der DD-Anweisung angegeben werden muß.

Folgende Parameter können oder müssen spezifiziert werden:

- Dateiname (optional):
  falls kein Name angegeben wird, ordnet das System der Datei einen eindeutigen Namen zu.
- Speichermedium, Datenträger:
  muß spezifiziert werden, falls es sich um eine neu zu erstellende Datei handelt;
  überflüssig, falls Datei katalogisiert ist.
- Im DCB-Parameter lassen sich die Eigenschaften der Datei spezifizieren.
- Disposition:
  Es wird spezifiziert, ob es sich um eine vorhandene alte Datei oder um eine neu zu erstellende Datei handelt, ob sie katalogisiert ist oder werden soll, ob sie aus dem Katalog entfernt werden soll, ob sie den ganzen Job oder nur für den laufenden Job-Step aufbewahrt werden soll.
- Ist eine neu zu erstellende Datei auf einem Speicher mit direkter Zugriffsmöglichkeit abzulegen, so müssen genaue Speicheranforderungen gemacht werden (Anzahl der Zylinder etc.).

Die DD-Anweisung ist äußerst nützlich bei Programmen, die in einer problemorientierten Programmiersprache (FORTRAN, COBOL, usw.) geschrieben sind, da die Dateieigenschaften nicht in der Quellsprache formuliert werden können.

./. 15

5.4 Die Beschreibung des Datensteuerblocks ist [2] S. 22 - 23, [4] S. 63 - 70, [7] S. 21 - 78, [11] S. 706 - 713 zu entnehmen.

6. Zugriffstechniken ([2] S. 23, [4] S. 71 - 75)

Es sind zwei allgemeine Techniken für den Transfer von Daten vorgesehen:

a) erweiterte Zugriffstechnik (queued access technique),
b) einfache Zugriffstechnik (basic access technique).

Die erweiterte Zugriffstechnik eignet sich in erster Linie für serielle Verarbeitung. Kontrollfunktionen bei der Durchführung von Ein/Ausgabe-Operationen werden vom System automatisch durchgeführt.

Die einfache Zugriffstechnik eignet sich in erster Linie für direkte Verarbeitung. Die Kontrolle bei der Durchführung von Ein/Ausgabe-Operationen obliegt weitgehend dem Benutzer.

6.1 Erweiterte Zugriffstechnik ([4] S. 71 - 72)

Makros für den Transfer zwischen Kern- und Hintergrundspeicher sind GET und PUT. Durch den GET-Makro wird dem Benutzer ein Satz zur Verfügung gestellt. Blocken und Entblocken wird vom System automatisch durchgeführt. Der GET-Makro hat vorwegnehmendes (anticipatory, lock-ahead) Puffern zur Folge, d.h. simultan dazu, daß dem Benutzer der gewünschte Satz zur Verfügung gestellt wird, werden leere Eingabepuffer gefüllt. Das System kontrolliert die richtige Abarbeitung der Puffer und sorgt für automatische Endebehandlung bei Ein/Ausgabe-Operationen. Das System sorgt für Überlappung von E/A-Operationen mit der CPU-Verarbeitung.

./. 16

Fehlerbehandlungen und Dateiende/Datenträgerende-Behandlungen werden automatisch durchgeführt. Aufgrund des vorwegnehmenden Pufferns eignet sich die erweiterte Zugriffstechnik hauptsächlich für serielle Verarbeitung.

6.2 Einfache Zugriffstechnik ([4] S. 72 - 74)

Der Datentransfer zwischen Kern- und Hintergrundspeicher geschieht mit Hilfe der Makros READ und WRITE. Durch READ/WRITE-Makros werden Blöcke und nicht Sätze transportiert. Blocken und Entblocken muß der Benutzer selbst vornehmen. Ein Puffer wird gefüllt oder geleert, wenn sich ein READ oder WRITE-Makro auf ihn bezieht. Vorwegnehmendes Puffern (vgl. 6.1) ist nicht möglich. READ und WRITE-Makros initialisieren lediglich E/A-Operationen, die korrekte Durchführung und Beendigung muß der Benutzer selbst überprüfen (CHECK-Makro). Das System übergibt nach Initialisierung eines READ/WRITE-Makros die Kontrolle dem Benutzer. Auf diese Weise ist Überlappung von E/A-Operationen mit der CPU-Verarbeitung möglich.

Die einfache Zugriffstechnik eignet sich vor allem dann, wenn die Sätze in mehr oder minder zufälliger Folge verarbeitet werden, d.h. bei direktem Zugriff.

6.3 Zugriffsmethoden ([4] S. 74 - 75, [11] S. 713 - 744)

Eine Zugriffsmethode ergibt sich aus einer Kombination zwischen einer Zugriffstechnik und einer verwendeten Speicherungsform (vgl. 7). Die verwendete Zugriffsmethode für eine Datei wird im DCB-Makro (vgl. 5.1) oder mit Hilfe der DD-Anweisung (vgl. 5.3) spezifiziert. Nachfolgende Tabelle gibt einen Überblick über die möglichen Zugriffsmethoden.

./. 17

Speicherungsform der Datei	Zugriffstechnik	
	Einfache	Erweiterte
starr fortlaufend	B S A M	Q S A M
logisch fortlaufend	B I S A M	Q I S A M
gegliedert	B P A M	-
direkt	B D A M	-

Abb. 8: Zugriffsmethoden

7. Speicherungsformen einer Datei ([2] S. 27 - 30, [4] S. 88-118, [11] S. 263 - 271)

Es sind für die Organisation von Dateien vier verschiedene Speicherungsformen vorgesehen:

- starr fortlaufend (physical sequential),
- logisch fortlaufend (indexed sequential),
- gegliedert (partitioned),
- direkt (direct).

7.1 Starr fortlaufende Speicherungsform ([4] S. 88 - 96, [11] S. 263)

Die Speicherungsform heißt starr fortlaufend, wenn bei gegebenem Satz der logisch folgende Satz auch auf dem physikalisch folgenden Speicherplatz steht.

Auf Magnetband, Drucker, Kartenleser/stanzer, Lochstreifenleser/stanzer ist nur die starr fortlaufende Speicherungsform möglich.

Detaillierte Angaben bezüglich Verarbeitung und Erstellung einer starr fortlaufend gespeicherten Datei sind [4] S. 88 - 96 zu entnehmen.

./. 18

7.2 Logisch fortlaufende Speicherungsform ([4] S. 104 - 114, [11] S. 265 - 271)

Die Speicherungsform heißt logisch fortlaufend, wenn

a) jeder Satz mit einem Schlüssel versehen ist (dient der Identifizierung des Satzes),

b) bei gegebenem Satz der logisch nächste Satz durch den nächsthöheren Schlüssel bestimmt ist.

Als Speichermedien kommen nur Datenträger mit direkter Zugriffsmöglichkeit in Frage.

Bei einer logisch fortlaufend gespeicherten Datei ist es möglich, Sätze zu ersetzen, einzufügen, zu löschen.

Um die Suchzeit für einen Satz zu verkürzen, wird beim Erstellen der Datei ein Inhaltsverzeichnis angelegt.

Beim Erstellen einer logisch fortlaufend gespeicherten Datei muß in der DD-Anweisung Speicherplatz für drei Bereiche beantragt werden:

- Hauptbereich (prime data area)
  dient im wesentlichen dazu, die Sätze beim Erstellen der Datei aufzunehmen;
- Überlaufbereich (overflow area)
  dient dazu, um Sätze, die dem Schlüssel nach auf einer Spur im Hauptbereich eingeordnet werden müßten, dort aber keinen Platz mehr finden, aufzunehmen;
- Bereich für das Inhaltsverzeichnis (index area)

7.2.1 Aufbau des Inhaltsverzeichnisses

Das Inhaltsverzeichnis ist auf verschiedenen Stufen organisiert. Beim Erstellen der Datei werden automatisch vom Daten-Management folgende zwei Stufen erzeugt:

./. 19

- für jeden Zylinder des Hauptbereichs genau ein Spureninhaltsverzeichnis (track index);
- für den gesamten Hauptbereich ein Zylinderinhaltsverzeichnis (cylinder index).

Auf Wunsch des Benutzers kann das Zylinderinhaltsverzeichnis selbst wieder untergliedert werden und mit Hilfe eines Hauptinhaltsverzeichnisses (master index) beschrieben werden. Von diesem Hauptinhaltsverzeichnis kann erneut ein Hauptinhaltsverzeichnis der nächst höheren Stufe angelegt werden. Es sind drei Stufen von Hauptinhaltsverzeichnissen möglich.

Das Spureninhaltsverzeichnis eines Zylinders steht auf der (den) ersten Spur(en) dieses Zylinders. Es enthält für jede Spur des Zylinders zwei Einträge, einen Normaleintrag und einen Überlaufeintrag. Der Normaleintrag enthält eine Spuradresse und den Schlüssel des letzten Satzes auf dieser Spur. Der Überlaufeintrag gibt Aufschluß darüber, ob Sätze, die logisch zu dieser Spur gehören, physikalisch auf einer Überlaufspur stehen.

Das Zylinderinhaltsverzeichnis enthält für jeden Zylinder des Hauptbereichs genau einen Eintrag. Dieser Eintrag besteht aus der Adresse des Spureninhaltsverzeichnisses dieses Zylinders und des höchsten Schlüssels in diesem Spureninhaltsverzeichnis.

Ein Eintrag in einem Hauptinhaltsverzeichnis (erster Stufe) verweist auf eine Spur im Zylinderinhaltsverzeichnis. Erstreckt sich ein Zylinderinhaltsverzeichnis über viele Spuren, so läßt sich für eine konstante Anzahl von Spuren jeweils ein Eintrag im Hauptinhaltsverzeichnis erzeugen.

./. 20

Die Struktur des Inhaltsverzeichnisses wird durch nachfolgendes Beispiel zusätzlich erläutert.

Eine logisch fortlaufend gespeicherte Datei benötige einen Hauptbereich von 12 Zylindern und einen Bereich von 3 Spuren für das Zylinderinhaltsverzeichnis. Weiter möge ein Hauptinhaltsverzeichnis existieren mit je einem Eintrag pro Spur des Zylinderinhaltsverzeichnisses.

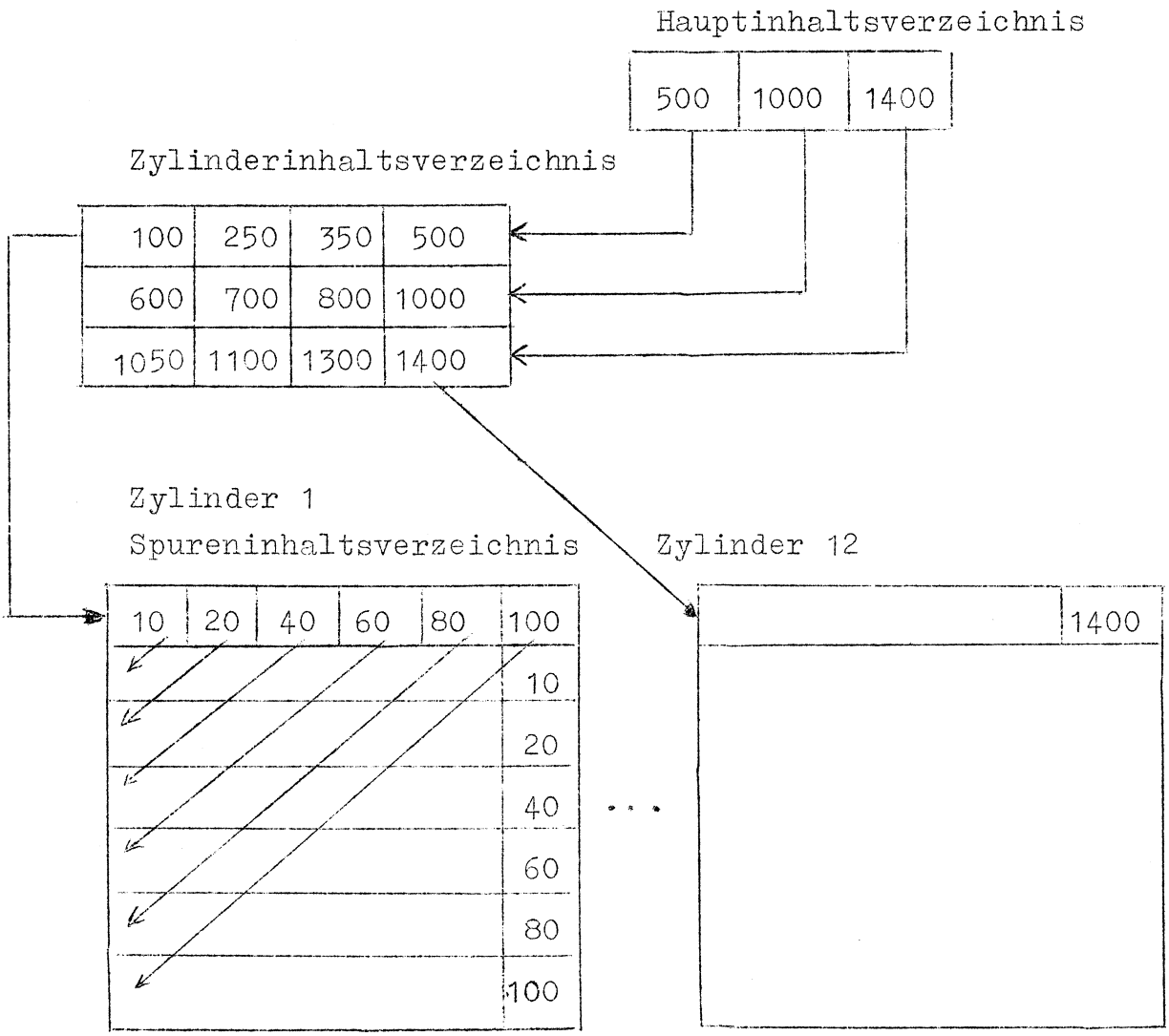


Abb. 9: Beispiel für das Inhaltsverzeichnis einer logisch fortlaufend gespeicherten Datei

./. 21

Ist ein Satz in eine logisch fortlaufend gespeicherte Datei einzufügen, so wird er gemäß seinem Schlüssel innerhalb einer Spur an die richtige Position gebracht. Sätze mit höheren Schlüsseln auf dieser Spur werden verschoben. Paßt der letzte Satz nicht mehr auf die Spur, so wird er auf eine Überlaufspur geschrieben und die Einträge im Spureninhaltsverzeichnis auf den neuesten Stand gebracht.

An dem Beispiel in Abb. 10 wird dieser Sachverhalt illustriert.

Im Hauptbereich einer logisch fortlaufend gespeicherten Datei können die Sätze geblockt oder ungeblockt geschrieben werden; im Überlaufbereich ist nur ungeblocktes Format möglich.

Beim Erstellen einer logisch fortlaufend gespeicherten Datei müssen die Sätze nach aufsteigenden Schlüsseln angeliefert werden.

Detaillierte Angaben bezüglich Verarbeitung und Erstellung einer logisch fortlaufend gespeicherten Datei sind [4] S. 104 - 114, 122 - 128 zu entnehmen.

./. 22

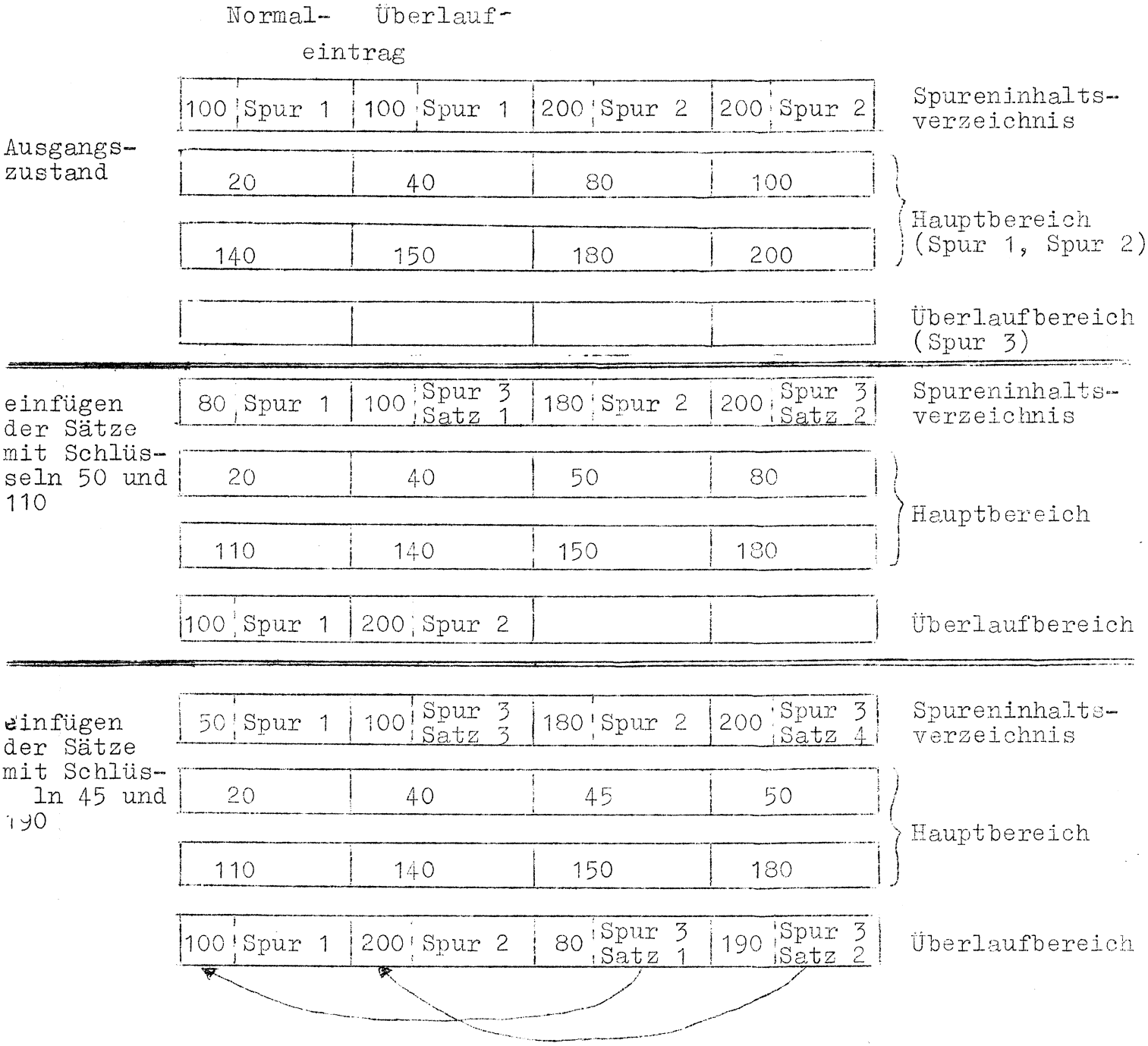


Abb. 10: Einfügen von Sätzen

./. 23

7.3 Gegliederte Speicherungsform ([4] S. 96 - 104, 121 - 122)

Die Speicherungsform heißt gegliedert, wenn die Datei aus endlich vielen Gliedern besteht. Ein Glied besteht aus starr fortlaufend organisierten Sätzen und hat einen einfachen Namen, der in einem zur Datei gehörenden Inhaltsverzeichnis (directory) gespeichert ist.

Als Speichermedien kommen nur Datenträger mit direkter Zugriffsmöglichkeit in Frage.

Es ist möglich, einzelne Glieder zu löschen bzw. hinzuzufügen. Löschen eines Glieds bedeutet löschen des Gliednamens im Inhaltsverzeichnis. Der von dem Glied beanspruchte Speicherplatz kann nicht zur Speicherung eines neuen Glieds verwendet werden. Hinzufügen eines Glieds bedeutet sequentielles anfügen der Sätze des Glieds und einfügen des Gliednamens gemäß der alphanumerischen Sortierreihenfolge in das Inhaltsverzeichnis.

Das Inhaltsverzeichnis steht am Anfang des von der Datei beantragten Speicherraums. Ein Eintrag im Inhaltsverzeichnis enthält den Namen eines Glieds und die Anfangsadresse des Glieds. Die Einträge sind nach aufsteigenden Gliednamen geordnet.

Nachstehende Abbildung erläutert die Gesamtorganisation einer gegliederten Datei:

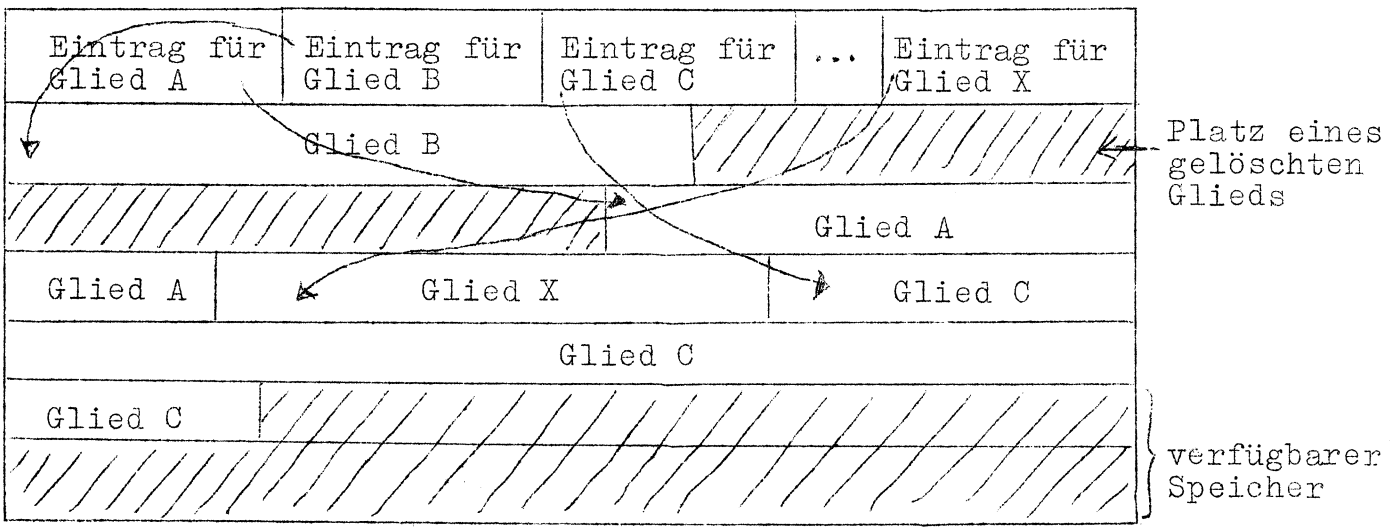


Abb. 11: Aufbau einer gegliederten Datei

./. 24

Das Erstellen einer gegliederten Datei stimmt weitgehend mit dem Erstellen einer starr fortlaufend gespeicherten Datei überein.

Gegliederte Dateien finden hauptsächlich bei der Speicherung von Programmbibliotheken Verwendung.

Detaillierte Angaben bezüglich Verarbeitung und Erstellung einer gegliederten Datei sind [4] S. 96 - 104, 121 - 122 zu entnehmen.

7.4 Direkte Speicherungsform ([4] S. 114 - 118, [11] 263 - 265)

Die Speicherungsform heißt direkt, wenn zwischen einem Satz und seinem Speicherplatz eine dem Benutzer bekannte Beziehung besteht. Diese Beziehung erlaubt es, Zugriff zu einem Satz zu bekommen, ohne ein Inhaltsverzeichnis durchsuchen zu müssen. Die Organisation der Datei wird vollständig vom Benutzer festgelegt.

Als Speicherungsmedien kommen nur Datenträger mit direkter Zugriffsmöglichkeit in Frage.

Die Sätze können nur in ungeblocktem Format, jedoch mit oder ohne Schlüsselfeld abgelegt werden.

Sätze lassen sich durch folgende Adressierungsmechanismen ansteuern:

- relative Satznummer,
- relative Spurnummer und Satznummer,
- relative Spurnummer und aktueller Schlüssel
  es wird ausgehend von der relativen Spurnummer nach einem Satz mit dem aktuellen Schlüssel gesucht;
- absolute Adresse

./. 25

8. Puffertechniken ([4] S. 78 - 88, [11] S. 713 - 715, 730 - 731)

Die Datenübertragung zwischen Hauptspeicher und E/A-Geräten geschieht mittels E/A-Operationen. Hauptspeicherbereiche, die auszugebende Daten enthalten, bzw. der Aufnahme einzugebender Daten dienen, heißen Puffer. Die Größe eines Puffers stimmt im wesentlichen mit der maximalen Blocklänge der ein- bzw. auszugebenden Datei überein. Bei der erweiterten Zugriffstechnik bezieht sich ein Verweis auf einen Puffer auf den nächsten Satz, d.h. auf ein Puffersegment.

Es ist möglich, einer Datei mehrere Puffer, d.h. einen Pufferpool, zuzuordnen.

8.1 Konstruktion von Puffern

Die Konstruktion eines Pufferpools kann auf drei verschiedene Weisen bewerkstelligt werden:

- statisch: Wenn beim Niederschreiben des Assemblerprogramms sowohl Anzahl als auch Größe der von einer Datei benötigten Puffer bekannt ist, kann ein entsprechender Hauptspeicherbereich reserviert und mit Hilfe des BUILD-Makros in einen Pufferpool strukturiert werden;
- explizit: Während der Ausführung des Programms wird durch einen GETPOOL-Makro ein Pufferpool konstruiert. Diese Methode ist zweckmäßig, wenn Größe und Anzahl der benötigten Puffer erst zur Ausführungszeit bekannt ist;
- automatisch: Während der Eröffnung einer Datei (OPEN-Makro) wird vom System ein Pufferpool konstruiert und der Datei zugeordnet. Dazu ist jedoch nötig, daß im Datensteuerblock dieser Datei der Wunsch nach einem Pufferpool spezifiziert wurde.

./. 26

Jeder Pufferpool, der einer Datei explizit durch den GETPOOL-Makro oder automatisch durch den OPEN-Makro zugeordnet wurde, muß vor Beendigung des Programms an das System zurückgegeben werden. Das geschieht mit dem FREEPOOL-Makro.

8.2 Pufferungsmethoden

Es gibt eine Vielzahl von Methoden wie Eingabepuffer, Ausgabepuffer und Verarbeitungsbereiche zueinander organisiert sind. Bei der Auswahl einer Pufferungsmethode spielt auch die Zugriffstechnik und die Speicherungsform einer Datei eine Rolle.

Die zwei wichtigsten Pufferungsmethoden sind:

- Einfache Pufferung (simple buffering): die Puffer dienen entweder als Eingabepuffer oder als Ausgabepuffer, können jedoch nicht für Eingabe und Ausgabe verwendet werden.
- Austauschende Pufferung (exchange buffering): die Puffer sind nicht notwendig einer einzigen Datei zugeordnet. Sie können als Eingabe- und Ausgabepuffer verwendet werden, was ein Transportieren der Sätze von einem Eingabe- zu einem Ausgabepuffer überflüssig macht.

Eine genaue Beschreibung der einzelnen Pufferungsmethoden ist [4] S. 81 - 88 zu entnehmen.

9. Einige kritische Anmerkungen zum Daten-Management

Das Daten-Management im OS/360 bietet dem Benutzer eine Vielfalt an Möglichkeiten und Erleichterungen, Daten zu organisieren und zu verarbeiten. Einige dieser Möglichkeiten, von denen es fragwürdig ist, ob sie für den Benutzer not-

./. 27

wendig sind, scheinen die Leistungsfähigkeit des Systems zu beeinträchtigen.

9.1 Der Benutzer hat die Möglichkeit, auf Speichern mit direktem Zugriff (Platte, Trommel, etc.) absolut zu adressieren und absolute Speicherbereiche anzufordern. Das zieht für das Gesamtsystem Konsequenzen nach sich:

a) Auf Plattenspeicher abgelegte Dateien dürfen zum Zwecke der Speicherbereinigung vom System nicht verschoben werden.

b) Eine logisch fortlaufend organisierte Datei kann nicht vom System selbst neu organisiert werden, was zweckmäßig wäre, um den Zugriff zu Überlaufspuren möglichst zu vermeiden.

c) Das Löschen einer Datei hat keine Speicherbereinigung zur Folge. Es wird lediglich der von der Datei beanspruchte Speicherraum freigegeben.

d) Obwohl mengenmäßig genügend Speicherplatz vorhanden sein kann, kann eine Datei nicht abgelegt werden, wenn nicht genügend große zusammenhängende Teile verfügbar sind.

9.2 Der Benutzer hat die Möglichkeit, Datenträgerkennsätze zu erstellen und insbesondere Datenträgernummern anzugeben. (Die Datenträgernummern dienen im Grunde dazu, eine eindeutige Identifizierung aller im System integrierten Datenträger zu gewährleisten.) Folgerungen daraus sind:

a) Ändert der Benutzer auf einem Plattenspeicher die Datenträgernummer, so muß er alle auf diesem Speicher katalogisierten Dateien neukatalogisieren. Tut er das nicht, so geht der Zugriff zu diesen Dateien mit Hilfe des Katalogs verloren.

./. 28

b) Das System kontrolliert nicht die eindeutige Zuordnung zwischen Datenträgern und Datenträgernummern.

Es erscheint zweckmäßig, wenn die Verwaltung und Vergabe von Datenträgernummern vom System gehandhabt wird.

9.3 Das Ansteuern einer Datei unter Angabe von Dateinamen und Datenträgernummer geschieht in folgenden drei Schritten:

1) Lesen des Datenträgerkennsatzes und überprüfen der Datenträgernummer auf Übereinstimmung.

2) Durchsuchen des Datenträgerinhaltsverzeichnisses nach dem gewünschten Dateikennsatz.

3) Positionieren auf den Datenteil der Datei.

Das Ansteuern einer katalogisierten Datei geschieht dadurch, daß der Katalog auf den Dateinamen durchsucht wird, um die zugehörige Datenträgernummer zu finden. Der restliche Ansteuerungsvorgang geschieht wie oben.

Der Nachteil dieses Ansteuerungsmechanismus ist:
Das mehrfache Positionieren des Plattenspeichers ist erheblich zeitraubend und beinhaltet bei Dateien, zu denen häufig zugegriffen werden muß, einen kaum vertretbaren Zeitverlust.

9.4 Es ist möglich, eine in einem Job-Step 1 verwendete Datei A dem nächsten Job Step 2 zu übergeben (DISP = (, PASS) in der DD-Anweisung). Die Ansteuerung von Datei A bei Verwendung in Job Step 2 geschieht jedoch nach dem in 9.3 beschriebenen aufwendigen Mechanismus. Diese zeitraubende Ansteuerung könnte weitgehend vermieden werden, wenn die in Job Step 1 verfügbare Information über den (die) Dateikennsatz(e) der Datei A während der Ausführung von Job Step 2 noch im Kernspeicher verfügbar wäre.

./. 29

10. Literaturverzeichnis

[1] IBM System/360 Operating System: Introduction,
Form C28 - 6534
Vorkenntnisse: keine
Ergänzungen: [4] - [10]

[2] IBM System/360 Operating System: Concepts and Facilities,
Form C28 - 6535
Vorkenntnisse: keine
Ergänzungen: [4] - [10]

[3] IBM System/360 Operating System: Assembler Language,
Form C28 - 6514
Vorkenntnisse: keine
Ergänzungen: [4] - [10]

[4] IBM System/360 Operating System: Supervisor and Data Management Services,
Form C28 - 6646
Vorkenntnisse: [2], [3]
Ergänzungen: [5] - [10]

[5] IBM System/360 Operating System: Supervisor and Data Management Macro-Instructions
Form C28 - 6647
Vorkenntnisse: [2], [3], [4]
Ergänzungen: [6] - [10]

[6] IBM System/360 Operating System: Job Control Language
Form C28 - 6539
Vorkenntnisse: [2], [3], [4]
Ergänzungen: [5], [8], [9], [10]

./. 30

[7] IBM System/360 Operating System: System Control Blocks
Form C28 - 6628
Vorkenntnisse: [1], [2], [4]
Ergänzungen: keine

[8] IBM System/360 Operating System: System Programmer's Guide
Form C28 - 6550
Vorkenntnisse: [2] - [7]
Ergänzungen: [9], [10]

[9] IBM System/360 Operating System: Utilities
Form C28 - 6586
Vorkenntnisse: [2], [3], [4],
Ergänzungen: [5] - [8]

[10] IBM System/360 Operating System: Storage Estimates
Form C28 - 6551
Vorkenntnisse: [2], [4], [6]
Ergänzungen: [6], [8]

[11] GERMAIN, Cl.B.: Das Programmierhandbuch der IBM/360
Carl Hanser Verlag, München, 1969
Vorkenntnisse: keine
Ergänzungen: IBM-Literatur