dietz 621

System-PASCAL

Erweiterung DFMS
Data File Management System
Benutzeranleitung

dietz 621

Betriebssystem XOS

Modul DFMS

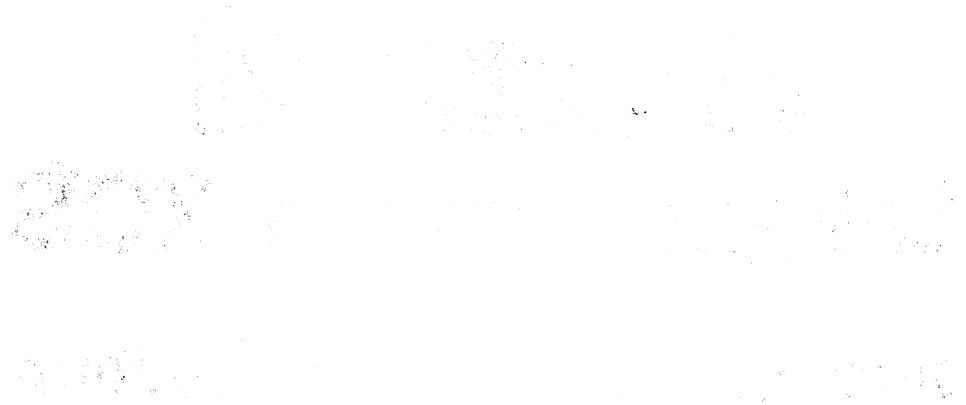
Data File Management System

Benutzeranleitung

DIETZ Computer SYSTEME

DIETZ Computer-Systeme · Heinrich Dietz · Solinger Straße 9 · 4330 Mülheim a. d. Ruhr · Telefon (0208) 4434-1

2-8107-01-198 Schutzgebühr DM 17,50

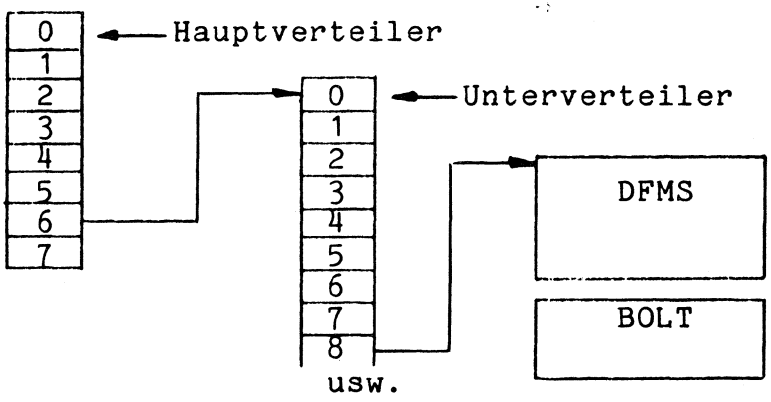


1. Allgemeines

1.1. Systemumfeld

Der Modul DFMS ist Teil des Betriebssystems XOS und gehört zur Gruppe 6 (CIOC = Complex Input Output Control). Der Modul DFMS kann durch den Modul BOLT erweitert werden. Während ohne BOLT-Verwaltung alle Funktionen nur "seriell reusable" sind, wird bei Verwendung des Moduls BOLT eine dateibezogene Verwaltung durchgeführt. Seriell reusable heißt, daß nur ein Teilnehmer den Modul zu einem bestimmten Zeitpunkt benutzen kann.

Während die Wartezeit bei den speicherresidenten Funktionen sehr klein ist und somit sich nicht störend auswirken kann, bedeutet der Aufruf einer Bibliotheksfunktion (z.B. Sortieren) eine Wartezeit, die auch in den Bereich von mehreren Minuten geht.
Eine Erweiterung durch die BOLT-Verwaltung macht den Modul DFMS reentrant,d.h. es können mehrere Benutzer gleichzeitig den Modul DFMS benutzen.
Dazu gehört allerdings auch die Vergabe von mehreren Transferpuffern (Anzahl der Teilnehmer = Anzahl der Puffer). Bei Vergabe nur eines Transferpuffers müssen die Teilnehmer sonst gegenseitig auf die Zuordnung des Puffers warten.



1.2. Einbindevorgang

Der Modul DFMS wird mit dem Hilfsprogramm "SERV,MODULE" in den Arbeitsspeicher eingelagert und über die Verteiler mit dem System verbunden. Über Zusatzangaben kann der Modul den Benutzerwünschen angepaßt werden.

```
SERV,MODULE,u,f,x,l,p,z,m
```

u : unit/Laufwerk
f : filename/Dateiname "MDFMS"
x : ab Release 5 des Betriebssystems XOS
ohne Bedeutung
l : Anzahl der Arbeitsnummern (16..255)
(Größe der OPEN-Tabelle)
p : ohne Bedeutung
z : Anzahl der Arbeitspuffer (0 oder 2..12)
z = 0: Ein Arbeitspuffer für
alle Benutzer
z ≠ 0: Anzahl der Arbeitspuffer gleich
Anzahl der Benutzer
m : Anpassungen an Benutzerwünsche
s.u.

Durch den letzten Parameter (m) kann der Modul in seiner Arbeitsweise entscheidend verändert werden. Besondere Vorsicht ist geboten, weil die Funktionsfähigkeit von Anwendungsprogrammen unter Umständen bei einer Änderung nicht mehr gewährleistet ist.

1.2.1. Anpassungen

Durch den letzten Parameter (m) können mehrere Anpassungen formuliert werden.
Der Parameter (m) setzt sich aus 4 Bits (Summanden) zusammen.

m := m5 + m4 + m3 + m2 + m1

m1 = 0

Das Sternchen (*) hat keine Sonderbedeutung, wenn es innerhalb des Vergleichsstrings bei der Funktion F11 vorkommt (s.a. Kap. 2.1.11 Seite 23).

m1 = 1

Ein oder mehrere Sternchen (*) innerhalb des Vergleichsstrings bei der Funktion F11 führen dazu, daß die entsprechenden Stellen beim Vergleich nicht berücksichtigt werden (s.a. 2.1.11 Suchen eines Indizes und Beispiel PDFS04).

m2 = 0 (Nur relevant für CBASIC)

Erkennt das System innerhalb der Argumentenliste die Folge "0: variable", oder ist der Wert der Variablen var1 bei der Folge "var1: variable" gleich null, so wird der aktuelle Wert des Satzzeigers in die hinter dem Doppelpunkt stehende Variable übertragen.

m2 = 2 (Nur relevant für CBASIC)

Bei der Folge "0: variable" bzw. wenn bei der Folge "var1: variable", der Wert von var1 gleich null ist, wird statt der aktuellen Satznummer das aktuelle Feld an die hinter dem Doppelpunkt stehende Variable übertragen.
(CBASIC Notation)

m3 = 0

Dateien werden benutzerabhängig geöffnet.

m3 = 4

Beim Öffnen von Dateien erfolgt keine Prüfung auf Benutzerzuordnung. Bei den Funktionen exklusives Öffnen (F3), Lesen und Sperren eines Satzes (F5) und Modifizieren und Freigabe eines Satzes (F6) wird der Zugriff auf die exklusiv geöffnete Datei bzw. der Zugriff auf den angewählten Satz nicht gesperrt. Vorsicht ist beim Schließen geboten. Wird die Datei von einem Benutzer geschlossen, ist sie für alle geschlossen.

m4 = 0

Der Modul DFMS bleibt "seriell reusable", d.h. die Benutzer können nur nacheinander den Modul benutzen. Ruft ein Benutzer den Modul auf, während der Modul schon benutzt wird, so muß dieser Benutzer warten.

m4 = 8

Der Modul DFMS soll gleichzeitig von mehreren Programmen aktiviert werden können. Es muß zusätzlich der Modul BOLT eingebunden werden.

m5 = 0

Suchverfahren von Indizes wie bei MDFMS - Rel. 15.0-15.3, physikalisch sequentiell.

m5 = 16

Suchverfahren von Indizes wie bei MDFMS - Rel. 15.4, physikalisch - logisch sequentiell.

1.3. Syntax der Aufrufe

1.3.1. Speicherresidente Funktionen

Der Aufruf erfolgt per Unterprogrammsprung zum Betriebssystem, allerdings werden die Funktionen vom Benutzerjob durchgeführt. Die gewünschte Funktion wird im Register R19 als Einbyte-Ganzzahl übergeben

1.3.1.1. Aufruf

```
LDC, ´19, n      /   n = Funktionsnummer
CSA, XKER, KERN  /   Unterprogrammsprung
                     (BS-Aufruf)
H, ´68           /   Modulgruppe CIOC (DFMS)
```

1.3.1.2. Liste der vorhandenen Funktionen

n	=	0	Arbeitsdatei erzeugen
		1	Indexdatei erzeugen
		2	Datei öffnen
		3	Datei exklusiv öffnen
		4	Datei schließen
		5	Lesen und Sperren des angewählten Satzes
		6	Modifizieren des angewählten Satzes
		7	Lesen des angewählten Satzes
		8	Beschreiben des angewählten Satzes
		9	Satzanwahl direkt oder indirekt
		10	Lesen des angewählten Indizes
		11	Suchen eines Indizes
		12	Eintragen eines Indizes
		13	Verknüpfen zweier Indizes
		14	Löschen eines Indizes
		15	Umbenennen eines Indizes

1.3.1.3. Modulbeschreibung

Parameterübergabe	: Register Ra und folgende
Registerbelegung	: Ra - R4F
Modulgröße	: ca. 4 KByte
Belegungssemaphore	: '24
Signale	: '22 und '23

1.3.1.4. Fehlermeldungen in DKER (Register R7E)

100	Dateiende bei Zugriff zur DFMS-Datei erreicht
101	Satzende bei Zugriff zur DFMS-Datei erreicht
102	Falscher Dateityp angegeben
103	Schlüssel nicht existent, doppelt (Code 2,6) oder leer
104	Dateigrößenparameter unzulässig
105	Arbeitsnummern alle belegt oder unzulässiger Wert

1.3.2. Bibliotheksfunktionen (Operatoren)

Der Aufruf erfolgt per Unterprogrammsprung zum eingebundenen Operator. Die Parameter müssen in den Registern R40 - R4F übergeben werden.

1.3.2.1. Aufruf

```
Lade R40 - R4F
CS*, '7C, anfang
```

* = Adressierungsart

1.3.2.2. Liste der Operatoren

Sortieren einer Indexdatei

Reorganisieren einer Indexdatei

Invertieren einer Satzdatei

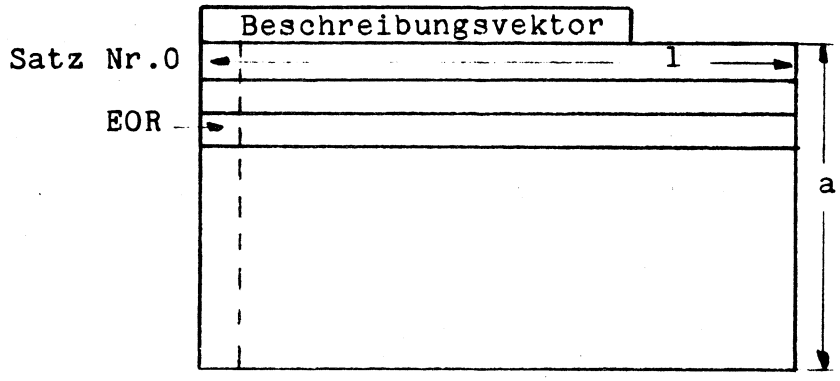
Reorganisieren einer Satzdatei/Indexdatei

Alle Operatoren bis auf "Reorganisieren einer Satzdatei/Indexdatei" müssen in Überlagerungstechnik benutzt werden, da ein Teil des Kodebereiches als Puffer benutzt wird und die Operatoren damit nur einmal ausgeführt werden können.

1.4. Struktur der DFMS-Dateien

1.4.1. Satzdatei

Die DFMS-Satzdatei hat eine feste Satzlänge und eine feste Satzanzahl. Ein Überlaufbereich ist nicht vorgesehen. Die beiden ersten Byte eines jeden Satzes enthalten den End of Record Zeiger (EOR). Der EOR zeigt die Füllhöhe des Satzes an.



EOR : End of Record / Füllhöhenzeiger, 2 Byte
l : physikalische Satzlänge (1=< l =< 32765)
a : Anzahl Sätze (1=< a =< 32767)

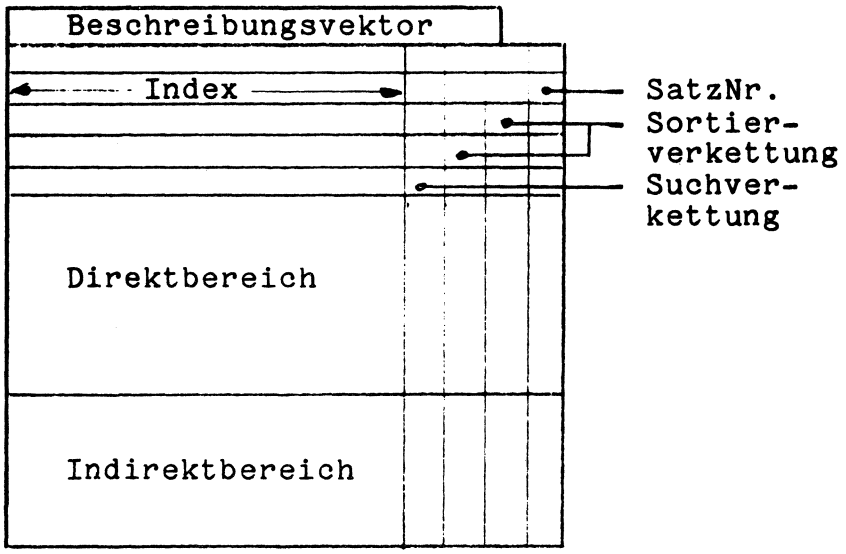
Der erste Satz hat die Satznummer 0.

Beschreibungsvektor (256 Byte)

Byte Nr. 0	: 'OX (X = Releasekennung)
1,2	: Anzahl Sätze
2,3	: Satzlänge + 2
5,6	: 0
7,8	: 0
9,10	: 2
11,12	: z.Zt. frei
13,14	: Freizeiger (FZ) für die Funktion F12
15-255	: z.Zt. frei

1.4.2. Indexdatei

Die Indexdatei ist in einen Direkt- und in einen Indirektbereich aufgeteilt. Der Direktbereich ist wiederum in Blöcke zu je 8 Einträgen aufgeteilt.



Beschreibungsvektor (256 Byte)

Byte Nr.	0	: 'CX (X = Releasekennung)
	1,2	: Anzahl Indizes
	2,3	: Indexlänge + 8
	5,6	: logisch erster Index
	7,8	: 0
	9,10	: Blockanzahl und Dateikennung
	11,12	: erster freier Platz im Indirektbereich
	13,14	: 0
	15-255	: z.Zt. frei

2. Semantik der Aufrufe

2.1. Speicherresidente Funktionen

2.1.1. Erzeugen einer Satzdatei

Funktion	:	R19	: 0	
Parameter	:	Rä	: u	unit/Laufwerk
		R03-08	: f	filename/Dateiname Satzdatei
		R09/0A	: sa	Anzahl der Sätze (=<32767)
		R0B/0C	: sl	Länge des Satzes (=<32765)

Bei dieser Funktion wird eine DFMS-Satzdatei angelegt. Die ersten 20 Byte enthalten die Dateibeschreibung, die übrigen werden auf '00 gesetzt.

Die Bedeutung der Dateibeschreibung kann dem Kap. 1.4.1 entnommen werden. Ebenso ist dort die Struktur der Datei erläutert.

2.1.2. Erzeugen einer Indexdatei

Funktion	:	R19	: 1	
Parameter	:	Ra	: u	unit/Laufwerk
		R03-08	: f	filename/Dateiname Indexdatei
		R09/0A	: ia	Anzahl der Indizes (=<32767)
		R0B/0C	: il	Länge des Indizes (=<119)
		R0D	: h	Hashverfahren

Diese Funktion erzeugt und formatiert eine Indexdatei. Sie enthält ebenfalls 20 Byte Dateibeschreibung

Im linken Halbbyte des Registers R0D wird eine Angabe über die Art des Zugriffverfahrens gemacht.

Die Datei wird in einen Direkt- und in einen Indirektbereich aufgeteilt. Der Direktbereich ist wiederum in eine Anzahl Blöcke aufgeteilt, die über eine Blockadresse das Eintragen der Indizes ermöglichen.
Die Indizes eines Blockes sind über den Suchzeiger miteinander verkettet. Ebenso verkettet der Blockzeiger die Indizes mit dem Block, die aufgrund eines Blocküberlaufes in den Indirektbereich eingetragen wurden.

2.1.2.1. Zugriffsverfahren

Die Bedeutung des linken Halbbytes ergibt sich aus der Summation der Bitwertigkeiten.

linkes Halbbyte	128	64	32	16

0 = sortiert; doppelte Indizes zugelassen
32 = sortiert; doppelte Indizes nicht zugelassen
64 = nicht sortiert; doppelte Indizes zugelassen
96 = nicht sortiert; doppelte Indizes nicht zugelassen

Die Bedeutung der Dateibeschreibung kann dem Kap. 1.4.2 entnommen werden.

Ebenso ist dort die Struktur der Datei erläutert.

2.1.3. Öffnen bzw. exclusives Öffnen einer Datei

Funktion	:	R19	:	2	Öffnen
		R19	:	3	Exklusives Öffnen
Parameter	:	Ra	:	u	unit/Laufwerk
			:	-1	s.u.
		R03-08	:	f	filename/Dateiname Satzdatei oder Indexdatei
		R09	:	u	unit/Laufwerk
			:	-1	s.u.
		R0A-0F	:	fi	filename/Dateiname Indexdatei
Rückgabe	:	Ra /03	:	w	Arbeitsnummer

Mit der Funktion F2/F3 wird eine Satzdatei oder eine Indexdatei oder eine Satzdatei verkettet mit einer Indexdatei geöffnet.
Im Falle, daß entweder nur eine Satzdatei bzw. nur eine Indexdatei geöffnet werden soll, muß im Register R09 der Wert ´FF übergeben werden.
Damit ergeben sich zwei Zugriffspfade für DFMS-Sätze.

a) direkt (Öffnen einer Satzdatei oder einer Indexdatei einzeln)
b) indirekt (Öffnen einer Satzdatei zusammen mit einer Indexdatei)

Wird die Funktion F3 (exclusives Öffnen) aufgerufen, so wird geprüft, ob die Datei bereits geöffnet ist. Falls ja, wird der rufende Teilnehmer in den Wartezustand versetzt (Wait signal ´22). Falls die Datei noch nicht geöffnet ist, wird der Zugang zu ihr für andere Teilnehmer gesperrt, bis die Funktion F4 aufgerufen wird.
Beim Öffnen bzw. exclusiven Öffnen werden die Dateiparameter, die für den Zugriff von Bedeutung sind, in der OPEN-Tabelle vermerkt (20 Byte). Verkettetes Öffnen (Satzdatei und Indexdatei) belegt zwei Arbeitsnummern. Bei Ra = -1 wird der Pufferinhalt gelöscht.

2.1.4. Schließen einer Datei

Funktion	:	R19	:	4
Parameter	:	R0 /03	:	w Arbeitsnummer
			:	-1 s.u.

Eine Datei schließen heißt, daß die zugriffsrelevanten Angaben in der Opentabelle gelöscht werden. Damit wird ein Eintrag - wenn nur eine Datei unter dieser Arbeitsnummer geöffnet wurde - wieder frei, bzw. werden zwei Einträge frei, wenn ein verkettetes Öffnen vorherging.
Außerdem wird eine evtl. Satzbelegung durch die Funktion F5 bzw. eine exclusive Dateibelegung durch die Funktion F3 aufgehoben.

Wenn in den Registern R0 -R03 eine (-1) 'FFFF übergeben wird, so werden alle Einträge gelöscht, die der rufende Benutzer bisher gemacht hat.

!! Besondere Vorsicht ist geboten, wenn der Modul DFMS in der Form eingebunden wurde, daß Dateien "Benutzerunabhängig" geöffnet werden. Wird die Datei von einem Benutzer geschlossen, so ist sie für alle geschlossen.

2.1.5. Lesen und Sperren eines Satzes

Funktion	:	R19	:	5
Parameter	:	R0 /03	:	w Arbeitsnummer
		R04/05	:	va Adresse der Variablenliste
		R06	:	anz Anzahl der Variablen
		R07	:	'80 mit Fortschalten des Satz- und/oder Indexzeigers
				'00 ohne Fortschalten des Satz- und/oder Indexzeigers.

Die Funktion F5 prüft, ob der zu lesende Satz gesperrt ist. Dies wäre der Fall, wenn ein anderer Benutzer die Funktion F5 oder F6 mit derselben Datei- und Satznummer aufgerufen hat. Falls ja, wird der rufende Benutzer in einen Wartestand versetzt.

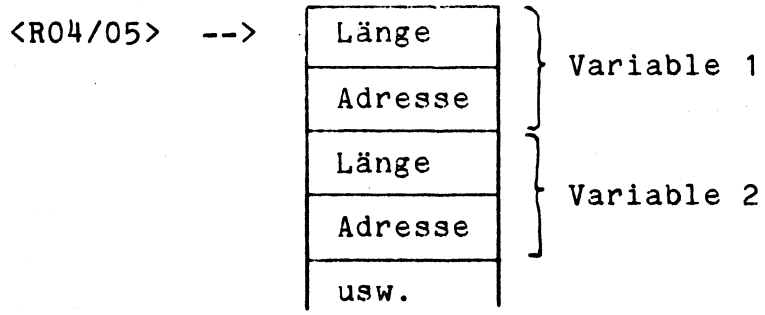
Falls nein, laufen folgende Vorgänge ab:

a) Satz gegen weitere Zugriffe sperren

b) aktuellen Stand des Satzoffsetzeigers (SOZ) in die OPEN-Tabelle retten.
Der Wert wird für die Funktion F6 benötigt.
!! VORSICHT nur nach Satzanwahl steht der SOZ auf 0.

c) Daten aus dem aktuellen Satz ab Wert des Satzoffsetzeigers lesen und den Variablen zuweisen, die in der Variablenliste aufgeführt sind. Die Adresse der Variablenliste muß in den Registern R04/05 übergeben werden.
Die Anzahl der Variablen muß im Register R06 übergeben werden.

Aufbau der Variablenliste:



Je Variable benötigt die Variablenliste 4 Byte.
Die Länge gibt die Anzahl der zu übertragenden Byte an, die Adresse weist auf das 1. Byte der Folge.

!! Wird im Register R07 der Wert '80 übergeben, so würden nach dem Lesen der Satzzeiger, bzw. bei verkettetem öffnen erst der Indexzeiger und dann der Satzzeiger auf einen neuen Wert gesetzt.

!! VORSICHT, dadurch wird die Satzsperre wieder aufgehoben.

2.1.6. Modifizieren und Freigeben eines Satzes

Funktion	:	R19	:	6
Parameter	:	Ra /03	:	w Arbeitsnummer
		R04/05	:	va Adresse der Variablenliste
		R06	:	anz Anzahl der Variablen
		R07	:	'80 mit Fortschalten des Satz- und/oder Indexzeigers
				'00 ohne Fortschalten des Satz- und/oder Index-zeigers

Mit der Funktion F6 werden Daten in den aktuellen Satz geschrieben. Der aktuelle Satz ergibt sich aus dem Stand des Satzzeigers (SZ) in der OPEN-Tabelle.
Die Daten werden über die Variablenliste beschafft. Aufbau der Variablenliste kann dem Kap. 2.1.5 entnommen werden.
Ist beim Aufruf der Funktion der Inhalt des Registers 06 (Anzahl der Variablen) gleich 0, so wird der Satz gelöscht. Der EOR-Zeiger wird auf den logischen Satzanfang gesetzt (EOR=2). Vor Ablauf der Funktion wird - ebenso wie bei Funktion F5 - gerüft, ob der Satz von einem anderen Benutzer gesperrt worden ist. Falls ja, wird der rufende Benutzer in einen Wartezustand versetzt.
Falls nein, laufen folgende Vorgänge ab:

a) Sperren des Satzes

b) Der Satzoffsetzeiger (SOZ) wird mit dem Wert der OPEN-Tabelle geladen. (Normalerweise werden die Plätze durch einen Aufruf der Funktion F5 vorher geladen (s.a. 2.1.5))

c) Übertragung der Daten

d) Satzoffsetzeiger -wie bei der Funktion F5- retten.

e) Falls der Satzoffsetzeiger (SOZ) durch die Inkrementierung beim Übertragen der Daten einen höheren Wert als der End of Record Zeiger (EOR) des aktuellen Satzes hat, wird der Satzoffsetzeiger auf den End of Record (EOR) Zeiger übertragen

2.1.7. Lesen eines Satzes

Funktion	:	R19	:	7
Parameter	:	Ra /03	:	w Arbeitsnummer
		R04/05	:	va Adresse der Variablenliste
		R06	:	anz Anzahl der Variablen
		R07	:	'80 mit Fortschalten des Satz- und/oder Indexzeigers
				'00 ohne Fortschaltung des Satz- und/oder Indexzeigers

Die Funktion F7 überträgt ohne Prüfung auf Satzsperre die Daten des aktuellen Satzes in die durch die Variablenliste spezifizierten Speicherbereiche. Es wird ab Stand des SOZ (Satzoffsetzeiger) übertragen. Wenn bei der Inkrementierung des Satzoffsetzeigers dieser einen höheren Wert als der EOR des aktuellen Satzes (Füllhöhe des angewählten Satzes) annimmt, wird Fehler 101 gemeldet.
Ist der in Register R07 vorgefundene Wert gleich '80, wird der Satzzeiger bzw. der Indexzeiger inkrementiert und der Satzoffsetzeiger auf 0 gesetzt, d.h. auf Satzanfang gestellt.

2.1.8. Schreiben eines Satzes

Funktion	:	R19	:	8
Parameter	:	R0 /03	:	w Arbeitsnummer
		R04/05	:	va Adresse der Variablenliste
		R06	:	anz Anzahl der Variablen
		R07	:	'80 mit Fortschalten des Satz- und/oder Indexzeigers
				'00 ohne Fortschalten des Satz- und/oder Indexzeigers

Die Funktion F8 überträgt ohne Prüfung auf Satzsperre die Daten, die durch die Variablenliste (s.a. 2.1.5 Funktion F5) spezifiziert sind, in den aktuellen Satz.
Vor der Übertragung wird der Satzoffsetzeiger (SOZ) mit dem Wert des EOR<SZ> geladen. Der EOR<SZ> enthält die Füllhöhe des aktuellen Satzes.
Somit kann ein Aufruf der Funktion F8 nur am logischen Satzende weiterschreiben. Nach Übertragung der Daten wird der Satzoffsetzeiger in den EOR<SZ> Zeiger übertragen.

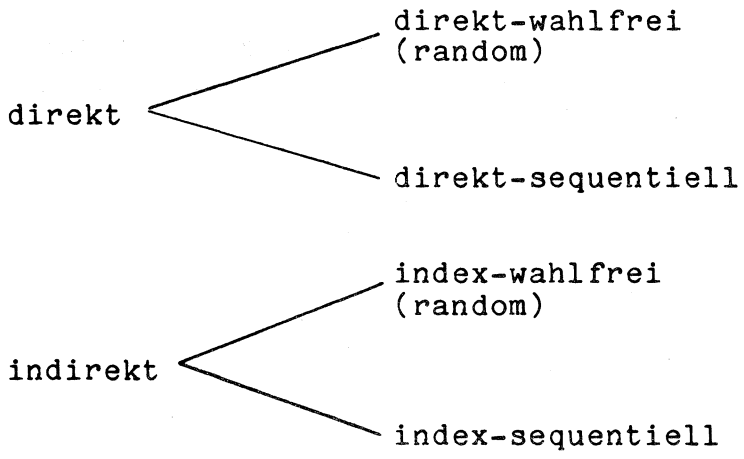
Nimmt der Satzoffsetzeiger (SOZ) einen höheren Wert als die physikalische Satzlänge an, so wird Fehler 101 gemeldet.

2.1.9. Anwahl eines Satzes

Funktion	:	R19	:	9
Parameter	:	Ra /03	:	w Arbeitsnummer
		R04/05	:	il Länge der Indexvariablen oder 0
		R06/07	:	ia Adresse der Indexvariablen oder Satznummer

Die Funktion F9 wählt entweder einen Satz direkt über die Satznummer (Übergabe des Wertes 0 in den Registern R04/05) oder indirekt über einen Index an. Der Satzzeiger der Satzdatei bzw. der Indexzeiger bei einer Indexdatei und einer verketteten Datei wird entsprechend den Inhalten der Register R04-R07 gesetzt. Dabei wird der Satzoffsetzeiger (SOZ) auf 0 gestellt.

Dadurch ergeben sich folgende Zugriffswege :



2.1.10. Index und Satznummer lesen

Funktion	:	R19	:	10
Parameter	:	R0 /03	:	w Arbeitsnummer
		R04/05	:	il Länge der Indexvariablen (Zielvariable)
		R06/07	:	ia Adresse der Indexvariablen (Zielvariable)
		R08/09	:	snr Adresse der Satznummer (Zielvariable)
		R0A	:	'80 falls Indexfortschaltung

Der aktuelle Index wird der Variablen zugewiesen, die durch die Register R04-R07 bestimmt ist. Die zum Index gehörende Referenzangabe zur Satzdatei (Satznummer) wird in die durch die Register R08/R09 spezifizierte Variable übertragen.
Der aktuelle Index ist durch den Stand des Indexzeigers in der Opentabelle definiert.

Wird im Register R0A der Wert '80 übergeben, so wird der Indexzeiger nach dem Zugriff auf den logisch nächsten Index gestellt. Wenn die Sortierverkettung nicht gesetzt ist (s.a. 2.1.2.1 und 2.2.1) so gibt es keinen logischen Nachfolger und es wird beim nächsten Zugriff EOF (End of File; ERR 100) gemeldet.

2.1.11. Index suchen

Funktion	:	R19	:	11
Parameter	:	Ra /03	:	w Arbeitsnummer
		R04/05	:	il Länge der Indexvariablen (Suchbegriff)
		R06/07	:	ia Adresse der Indexvariablen (Suchbegriff)
		R08/09	:	il Länge der Indexvariablen (Zielvariable)
		R0A/0B	:	ia Adresse der Indexvariablen (Zielvariable)
		R0C	:	vgl Vergl. Operator
		R0E/0F	:	Adresse eines Arbeitspuffers, Indexlänge

Mit der Funktion F11 wird derjenige Index bestimmt, der dem Suchbegriff (- definiert durch die Inhalte der Register R04-R09 u. R0C -) am nächsten kommt. Ist der Aufruf erfolgreich, d.h. es konnte ein Index angewählt werden, so wird der gefundene Index in die durch die Register R08-R0B spezifizierte Zielvariable übertragen. Der Index- und Satzzeiger wird entsprechend gesetzt, so daß ein Zugriff zum referenzierten Satz erfolgen kann.
Im Index, der durch die Register R04-R07 definiert ist (Suchbegriff), dürfen Maskenzeichen "*" angegeben sein, falls das Einbinden des Moduls DFMS entsprechend erfolgt ist (s.a. 1.2.1).

Es sind folgende Vergleichsoperatoren erlaubt :

<R0C> = '01 --> "="

= '02 --> "<"

= '03 --> "<="

= '04 --> ">"

= '05 --> ">="

2.1.12. Index eintragen

Funktion	:	R19	:	12
Parameter	:	Ra /03	:	w Arbeitsnummer
		R04/05	:	il Länge der Indexvariablen
		R06/07	:	ia Adresse der Indexvariablen
		R08/09	:	snr Satznummer oder ´FFFF
		R0A	:	op Eintragsoperator
		R0E/0F	:	pa Adresse eines Arbeitspuffers

Mit der Funktion F12 wird der Index eingetragen, der durch die Inhalte der Register R04-R07 definiert ist. Der Eintrag erfolgt in die Indexdatei, die durch die Arbeitsnummer (w) im Register Ra /03 vorgegeben ist. Gleichzeitig wird dem Index die Satznummer zugeordnet, die durch den Inhalt der Register R08/09 vorgegeben ist. Falls dort der Wert ´FFFF übergeben wurde, wird die Satznummer aus der dazugehörenden Satzdatei bestimmt. Die dazugehörende Satzdatei ergibt sich durch das verkettete Öffnen. Die Satznummer wird danach um eins erhöht und als neuer Freizeiger (FZ) in die Satzdatei zurückgeschrieben.

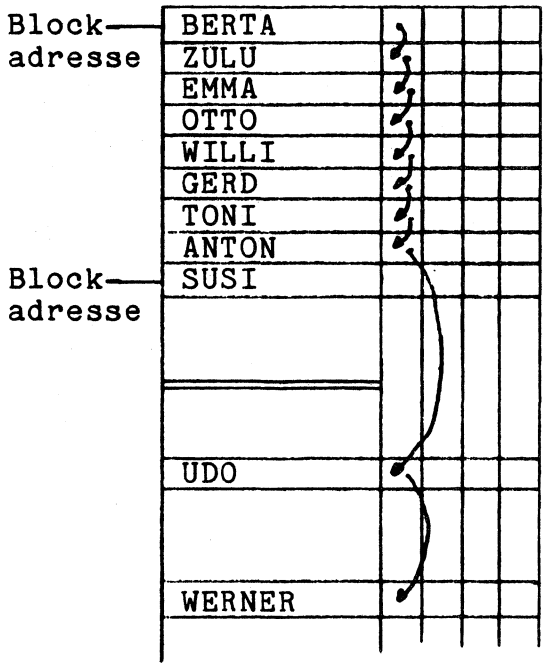
Es sind folgende Eintragsoperatoren erlaubt:

<R0A> = ´00 --> Eintrag abhängig vom Dateityp

= ´01 --> Eintrag sortiert

= ´02 --> Eintrag aufsteigend sortiert

Die Indexdatei ist in einen Direktbereich, der wiederum in eine Anzahl Blöcke aufgeteilt ist, und in einen Indirektbereich eingeteilt.
Der Block, in den der Index eingetragen ist, wird über eine HASH-Formel aus dem Index berechnet, Jeder Block faßt 8 Indizes. Da sich aufgrund der HASH-Formel Mehrdeutigkeiten ergeben, werden die Indizes eines Blockes über den HASH-Zeiger verkettet. Läuft ein Block über, so wird der nächste Index, der in diesen Block eingetragen werden soll, in den Indirektbereich geschrieben und über den HASH-Zeiger mit seinem Block verbunden.



Bei dem Beispiel wurde angenommen, daß alle eingetragenen Indizes aufgrund der HASH-Formel dieselbe Blockadresse lieferten. Die Sortierverkettung wird unabhängig davon über weitere zwei Zeiger vorgenommen.

Sind aufgrund des Typs der angewählten Indexdatei (s.a. 2.1.2) keine Doppeleintragungen erlaubt, so wird Fehler 103 gemeldet, wenn ein identischer Index beim Eintragen gefunden wird.

2.1.13. Indizes verknüpfen

Funktion	:	R19	:	13
Parameter	:	Ra /03	:	w1 Arbeitsnummer
		R04/05	:	il Länge der 1. Indexvariablen
		R06/07	:	ia Adresse der 1. Indexvariablen
		R08/09	:	w2 Arbeitsnummer
		R0A/0B	:	il Länge der 2. Indexvariablen
		R0C/0D	:	ia Adresse der 2. Indexvariablen
		R0E/0F	:	pa Adresse eines Arbeitspuffers Indexlänge + 8

Die Funktion F13 trägt den durch die Inhalte der Register Ra -R07 spezifizierten Index ein (s.a. 2.1.12). Die Satznummer wird durch den schon eingetragenen Index (R08-R0D) bestimmt. Wenn in R0A/R0B der Wert 'FFFF übergeben wird, so wird der aktuelle Eintrag unter der Arbeitsnummer w2 der Opentabelle benutzt, um die Satznummer zu bestimmen. Die Bestimmung der Satznummer über die Angabe eines Indizes in den Registern R08-R0D entspricht den Funktionen F11 (Anwahl) und F10 (Lese Index mit Satznummer). Die Übergabe des Wertes 'FFFF in den Registern R0A/R0B wird als "aktueller Index" bezeichnet.

!! Zu beachten ist, daß es bei den Indexdateien vom Typ 0 bzw. 32 möglich ist, doppelte Indizes einzutragen. Die Funktion F11 (Anwahl) ist nur in der Lage, den in der zeitlichen Reihenfolge als ersten eingetragenen Index anzuwählen. Sind also doppelte Indizes vorhanden, so kann nur über die Möglichkeit des "aktuellen Indizes" auf den zweiten und auf folgende Indizes zugegriffen werden.

2.1.14. Index löschen

Funktion	:	R19	: 14	
Parameter	:	R00 /03	: w	Arbeitsnummer
		R04/05	: il	Länge der Indexvariablen
		R06/07	: ia	Adresse der Indexvariablen
		R0E/0F	: pa	Adresse eines Arbeitspuffers Indexlänge + 8

Die Funktion F14 löscht den durch den Inhalt der Register R04-R07 definierten Index; es wird dabei das erste Zeichen des Indizes auf den Wert ´00 gesetzt. Der freiwerdende Platz ist erst nach einer Reorganistion der Indexdatei wieder besetzbar (s.a. 2.2.2).

!! Es ist unbedingt zu beachten, daß der Funktion F14 die Funktion F11 (Anwahl) vorgeschaltet wird, wenn ein Index angegeben wird.
Das bedeutet, daß bei Angabe eines Indizes immer der gelöscht wird, der in der zeitlichen Reihenfolge als erster eingetragen wurde.
Dieses kann durch die Übergabe des Wertes 0 in den Registern R06/07 umgangen werden. Es wird dann der sogenannte "aktuelle Index" gelöscht.

2.1.15. Index umbenennen

Funktion:	R19	: 15	
Parameter:	Ra /03	: w	Arbeitsnummer
	R04/05	: il1	Länge des alten Indizes
	R06/07	: ia1	Adresse des alten Indizes
	R08/09	: il2	Länge des neuen Indizes
	R0A/0B	: ia2	Adresse des neuen Indizes
	R0E/0F	: pa	Adresse eines Arbeitspuffers Indexlänge + 8

Auch beim Umbenennen von Indizes ist zu beachten, daß die Funktion F11 (Anwahl) vorgeschaltet wird, wenn nicht der "aktuelle Index" (il1 = 0) angegeben wird. Dadurch wird wieder der in der zeitlichen Reihenfolge als erster eingetragene Index umbenannt.

Zu beachten ist außerdem, daß der alte Index nicht einfach umbenannt werden kann, da ja die HASH-Formel über den Index die Blockadresse und damit den Platz bestimmt.
Ein Umbenennen erfolgt in der Form, daß die Satznummer des alten Indizes bestimmt und der Index dann gelöscht wird.
Der neue Index wird mit der Satznummer des alten Indizes eingetragen. Es ergeben sich somit bei der Funktion F15 ebenso Freiplätze wie bei der Funktion F14. Der freiwerdende Platz ist auch erst nach einer Reorganisation wieder besetzbar (s.a. 2.2.2).

2.2. Bibliotheksfunktionen

Die Funktionen sind bis auf "Reorganisieren einer Satz/Indexdatei" nicht reusable. Das bedeutet, daß die Funktionen vor jeder Ausführung vom Externspeicher geladen werden müssen (Überlagerungstechnik).
Auf dem Externspeicher (Systemresidenz) werden zusätzlich die Dateien SFILE0 und SFILE1 benötigt. Die Länge der beiden Dateien muß gleich sein und berechnet sich nach folgender Formel :

Je Index einer zu verarbeitenden Indexdatei wird die Indexlänge + 2, gerundet auf die nächste 2er Potenz, benötigt. Außerdem müssen die Dateien mindestens die physikalische Länge der zu bearbeitenden Indexdateien haben.

Folgende Bibliotheksfunktionen sind vorhanden :

- Sortieren einer Indexdatei
- Reorganisieren einer Indexdatei
- Invertieren einer Satzdatei
- Reorganisieren einer Satzdatei/Indexdatei

2.2.1. Sortieren einer Indexdatei

Aufruf: CS*,'7C,anfang

(* = Adressierungsart)

Parameter 1:	R40/41	: u	unit/Laufwerk
	R42/43	:	ohne Bedeutung
Parameter 2:	R44/45	: fl	Länge des Dateinamens
	R46/47	: fa	Adresse des Dateinamens

Der erste Parameter gibt das Laufwerk an. Der zweite Parameter bestimmt den Dateinamen (f = filename) der Indexdatei, die sortiert werden soll.
Für die Ausführung der Funktion werden die beiden Dateien SFILE0 und SFILE1 benötigt. Die Funktion ist nicht "reusable", d.h. sie muß in Überlagerungstechnik benutzt werden.
Nach Abarbeitung der Funktion ist die Indexdatei sortiert und kann für indexsequentielle Zugriffe verwendet werden.

2.2.2. Reorganisieren einer Indexdatei

Aufruf: CS*,'7C,anfang

(* = Adressierungsart)

Parameter 1:	R40/41	:	u	unit/Laufwerk
	R42/43	:		ohne Bedeutung
Parameter 2:	R44/45	:	fl	Länge des Dateinamens
	R46/47	:	fa	Adresse des Dateinamens
Parameter 3:	R48/49	:	u	unit/Laufwerk
	R4A/4B	:		ohne Bedeutung
Parameter 4:	R4C/4D	:	fl	Länge des Dateinamens
	R4E/4F	:	fa	Adresse des Dateinamens

Die beiden ersten Parameter geben die Quelldatei (das ist die Datei, die reorganisiert werden soll) an; die beiden letzten Parameter geben die Zieldatei an. Quelldatei und Zieldatei dürfen identisch sein, da die Reorganisation mit Hilfe der beiden Dateien SFILE0 und SFILE1 geschieht.
Quelldatei und Zieldatei werden immer dann verschiedene Dateien sein, wenn neben der Reorganisation auch eine Änderung des Verhaltens der Indexdatei durch eine Änderung des Typs erreicht werden soll. Nach Ablauf der Funktion sind die Freiplätze - die durch Löschen oder Umbenennen von Indizes entstanden sind - wieder verfügbar. Außerdem sind die Sortierzeiger gesetzt, sodaß eine indexsequentielle Verarbeitung im Anschluß möglich ist. Es werden die beiden Dateien SFILE0 und SFILE1 benötigt. Da die Funktion nicht "seriell reusable" ist, muß der Aufruf in Überlagerungstechnik erfolgen.

2.2.3. Invertieren einer Satzdatei

Aufruf: CS*,'7C,anfang

(* = Adressierungsart)

Parameter 1:	R40/41	:	u	unit/Laufwerk
	R42/43	:		ohne Bedeutung
Parameter 2:	R44/45	:	fl	Länge des Dateinamens
	R46/47	:	fa	Adresse des Dateinamens
Parameter 3:	R48/49	:	va	Adresse der Feldvariablen
	R4A/4B	:	vr	Restlänge der Feldvariablen
Parameter 4:	R4C/4D	:	w	Arbeitsnummer der Indexdatei
	R4E/4F	:		ohne Bedeutung

Die Funktion wird zum nachträglichen Invertieren einer Satzdatei benutzt. Die beiden ersten Parameter geben die Satzdatei an.
Der dritte Parameter definiert indirekt - d.h. über die Angabe eines Integerfeldes - die Satzteile, die den neuen Index bilden sollen. Jeweils zwei Zahlen des Feldes geben einen Abschnitt des Satzes an. Der neue Index kann aus beliebig vielen Abschnitten des Satzes aufgebaut werden. Die Reihe der Zahlenpaare wird durch den Wert -1 abgeschlossen.
Als 4. Parameter wird die Arbeitsnummer der Indexdatei angegeben, die die neuen Indizes aufnehmen soll.
Die Indexdatei ist "einfach" zu öffnen (s.a. 2.1.3).
Nach Durchführung dieser Funktion ist die angegebene Indexdatei sortiert.

2.2.4. Reorganisieren einer Satzdatei/Indexdatei

Aufruf: CS*,'7C,anfang

(* = Adressierungsart)

Parameter 1:	R40/41	:	u	unit/Laufwerk der Satz/Indexdatei
	R42/43	:		ohne Bedeutung
Parameter 2:	R44/45	:	fl	Länge des Dateinamens
	R46/47	:	fa	Adresse des Dateinamens
Parameter 3:	R48/49	:	u	unit/Laufwerk der Hilfsdatei
	R4A/4B	:		ohne Bedeutung
Parameter 4:	R4C/4D	:	fl	Länge des Dateinamens
	R4E/4F	:	fa	Adresse des Dateinamens

Sätze, die durch die Funktion F6 gelöscht wurden, können nur durch wahlfreien Zugriff, d.h. über die Angabe der Satznummer wieder beschrieben und verwendet werden. Sollen diese Sätze aber auch für Neueinträge mit F12 wieder zur Verfügung stehen, so muß die Satzdatei reorganisiert werden. Nach Ablauf der Funktion sind alle Sätze lückenlos in der Datei gespeichert und der Freizeiger (FZ) hat den Wert des ersten freien Satzes. Da durch diese Reorganisation alle Referenzangaben zu der Satzdatei - die eingetragenen Satznummern in den Indexdateien - falsch sind, muß die Reorganisationsfunktion auch auf alle zu der Satzdatei gehörenden Indexdateien angewendet werden.
Die durch die Parameter 3 und 4 angegebene Hilfsdatei wird vom System in der benötigten Größe selbständig angelegt und mit den alten Satznummern -vor der Reorganisation- bzw. mit den neuen Satznummern -nach der Reorganisation- gefüllt.
Dadurch ist es dem System möglich, die betroffenen Referenzangaben in den Indexdateien auf den richtigen Wert zu setzen.

3. Anhang

3.1. Verwendete Abkürzungen

Sachwort	Erläuterung, Seite etc.
a	Anzahl der Sätze
anz	Anzahl der zu übertragenden Variablen, auch Länge der Variablenliste.
BS	Betriebssystem
BS-Aufruf	Betriebssystem Aufruf (Funktionsaufruf)
CIOC	Modul des Betriebssystems XOS Modul Nr. 6 Complex Input / Output System
CS*	Unterprogrammsprung; die Adressierungsart ist an Stelle des * mit A, X oder L anzugeben.
CSA	Assemblerbefehl der Zentral-einheit DIETZ 621 Call Sub Absolut
DBMS	Data Based Management System steht z.B. über die Sprache MAGICS zur Verfügung
DFMS	Data File Management System
EOR	End of Record Satzende-Zeiger, zeigt die Füllhöhe eines Satzes an.

Sachwort	Erläuterung, Seite etc.
EOR<SZ>	End of Record des aktuellen Satzes. SZ = Satzzeiger.
FZ	Freizeiger, zeigt den ersten freien Satz einer Satzdatei an u. wird bei F12 inkrementiert.
f	filename/Dateiname
fa	Adresse des Dateinamens beim Operator Reorganisieren S. 34.
fi	filename/Dateiname einer Indexdatei
fl	Länge des Dateinamens, Operator Reorganisieren S.34
h	Hashverfahren h = 0/32/64/96
H	Assemblerbefehl der Zentraleinheit DIETZ 621, Speicherbelegung mit sedezimalem Wert.
ia	S. 13, Indexanzahl, max.:32767 S 21, Adresse der Indexvariablen.
il	S. 13, Indexlänge, max.:119 S, 21, Länge der Indexvariablen.
KERN	Einsprungadresse des BS für Funktionsaufrufe; KERN = '4020.
Kap	Kapitel
l	S. 4, Größe der Opentabelle, S. 10 , Satzlänge.

Sachwort	Erläuterung, Seite etc.
m	letzter Parameter bei SERV,MODULE auf S. 4.
n	S 13, Exponent zur Basis 2, um die Größe des Direktbereichs festzulegen.
Nr.	Nummer
op	Eintragsoperator für die Funktion F12; Index eintragen.
PDFS04	Name eines Beispielprogramms Pascal Data File System und eine laufende Nummer.
R19	Register 19 der Zentral-einheit DIETZ 621
Ra	Standardakkumulator der ZE DIETZ 621; Registeradresse ´02.
s.a	siehe auch
sa	Satzanzahl: bei DFMS-Sätzen maximal 32767.
sl	Satzlänge: bei DFMS-Sätzen maximal 32767.
snr	Satznummer, bei einem Zugriff zur Satzdatei wird der SZ auf den Wert der snr gesetzt.

Sachwort	Erläuterung, Seite etc.
S	Seite
SERV	Aufruf eines Hilfsprogramms
u	unit/Laufwerk
va	Adresse der Variablenliste
vgl	Vergleichsoperator für die Funktion F11,Indexsuchen.
w	worknumber/Arbeitesnummer wird vom System beim Öffnen vergeben.
x	zeigt einen Parameterplatz an, der ab Rel.5 des BS nicht mehr besetzt werden muß.
xd	Größe des Direktbereichs einer Indexdatei.
XKER	Rückkerregister für den Sprung zum BS; XKER = ´1E.
XOS	Betriebssystemfamilie der Rechnerserie 621.
z	Anzahl der Arbeitspuffer

3.2. Glossar

Sachwort	Erläuterung, Seite etc.
Aktueller Index	S. 23, S. 27, S. 28, S. 29, wird benötigt, um bei Mehrfacheintragungen Indizes zu finden.
Anpassung	S. 5, durch die Parameter des Einbindekommandos kann der Modul DFMS angepaßt werden.
Arbeitsnummer	S. 4 wird vom System beim "Öffnen" einer Datei vergeben.
Arbeitspuffer	S. 4, Anzahl ist beim Generieren festlegbar, wird vom Modul DFMS benutzt.
Argumentliste	S. 5, Liste der zu übertragenden Variablen beim Lesen oder Schreiben eines Satzes.
BOLT-Varaible	S. 3, durch den Modul MBOLT kann der Modul DFMS parallel genutzt werden.
Belegungssemaphore	S. 9, zeigt an, ob ein Betriebsmittel (Gerät, Datei, Programm) benutzt werden kann.
Benutzerabhängig	Öffnen: S. 6
Benutzerjob	Führt sowohl den Dialog als auch den Datentransfer von und zur Platte.(s.a. S. 7)
Benutzerabhängig	Öffnen: S.16, muß bei der Generierung des Systems angemeldet werden (s.a. S. 6).
Beschreibungsvektor	S. 10 ,insgesamt 256 Byte, von denen 20 Byte Daten enthalten.
Bibliotheksfunktionen	S. 9, werden auch Operatoren genannt (s.a. S. 30).

Sachwort	Erläuterung, Seite etc.
Blockadresse	S.13, wird nach einem festgelegten Algorithmus aus dem Index berechnet (s.a. S. 26).
Direkt-sequentiell	S. 22, Zugriff zu Sätzen einer DFMS-Datei mit autom. Inkrementierung des Satzzeigers.
Direkt-wahlfrei	S.22, Zugriff zu einem Satz unter Angabe der Satznummer auch "Random Zugriff" genannt.
Direktbereich	S. 11, Anteil der Indexdatei, der direkt und damit schnell zugeifbar ist (s.a. S. 14).
Doppelte Indizes	S. 14, eine Indexdatei kann doppelte Indizes enthalten. (s.a. S. 25)
Einbindevorgang	S. 4, wird durch SERV,MODULE ausgelöst. Anpassung an die Benutzerwünsche.
Einbyte-Ganzzahl	S. 7, spezifiziert die Funktion beim Auffruf des Moduls DFMS. Wert zwischen 0 und 255.
End of Record	S. 19, auch Füllhöhenanzeige eines Satzes genannt.
End of Record Zeiger	S.10, zeigt die Belegung eines Satzes an und wird auch als Füllhöhenzeiger bezeichnet.

Sachwort	Erläuterung, Seite etc.
Exklusives Öffnen	S. 15, bestimmte Arbeitesgänge (z.B. Reorganisieren) erfordern das exklusive Öffnen.
Fehlermeldungen	S. 8, werden im Register DKER übergeben; DKER='7E.
Freigeben	S. 19, ein Satz wird durch die Anwahl eines neuen Satzes freigegeben.
Freizeiger	S. 25, wird durch die Funktion F12 inkrementiert, bzw durch SATZREORG gesetzt (s.a. S. 34).
Füllhöhenzeiger	S. 10, zeigt an, wie weit ein Satz beschrieben ist. Wird auch End of Record genannt.
Funktionsnummer	S. 7, definiert die Funktion beim Aufruf des Moduls DFMS. Wert zwischen 0 und 255.
HASH-Formel	S. 26, auch Suchalgorithmus genannt. Wird benutzt, um den Platz eines Indizes zu berechnen.
Hauptverteiler	Liste des BS, in der alle Moduln des Systems eingetragen sind.
Hilfsdatei	S. 24, wird durch das System auotm. bei Satzdatei Reorg. angelegt.
Inedx	wird auch Schlüssel oder Ordnungsbegriff genannt.

Sachwort	Erläuterung, Seite etc.
Index-sequentiell	S. 22, Zugriff zu Sätzen oder Indizes mit autom. Inkrementierung des Index/Satzzeigers.
Index-wahlfrei	S. 22, Zugriff zu einem Satz oder auch zu einem Ordnungsbegriff über einen Index.
Indexdatei	S. 11, dient zur Aufnahme der Ordnungsbegriffe (s.a. S. 13).
Indexdatei Reorg.	S. 34, Aktualisieren der Satznummern (Referenzen) einer Indexdatei nach Satzdatei Reorg.
Indexzeiger	S. 23, zeigt nach dem Öffnen einer datei auf den kleinsten, sonst auf den aktuellen Index.
Indirekt	S. 22, wahlfreier oder sequentieller Zugriff über Indizes (Ordnungsbegriffe).
Indirektbereich	S. 11, nimmt die Indizes auf, die keinen Platz mehr im Direktbereich haben.
Invertieren	S. 33, nachträgliches Aufbauen einer Indexdatei anhand von Teilen einer Satzdatei.
Maskenzeichen	S. 24, es können "*" bei nur teilweise bekannten Indizes verwendet werden (s.a. S.3/4).
Modifizieren	S. 19, Ändern eines Satzes, Beschreiben eines Satzes ohne Rücksicht auf den EOR Zeiger.

Sachwort	Erläuterung, Seite etc.
Struktur	S. 10, physikalischer Aufbau einer Satz-oder Indexdatei.
Suchbegriff	S. 24, wahlfreier Zugriff über einen Index; der Suchbegriff kann Sternchen einthalten.
Suchverkettung	S. 11, wird bei jedem Eintrag eines Indizes gesetzt. Dient zur Verkettung aller Indizes.
Systemresidenz	S. 30, Nummer des Plattenlaufwerks, von dem das BS geladen wurde.
Überlagerungstechn.	S. 31, auch Segmentierung und Overlay genannt. Nachladen des Operators vor der Ausführung.
Unterverteiler	Liste des BS, in der alle Funktionen eines Moduls eingetragen sind.
Variablenliste	S. 18,enthält die Adressen und die Längenangaben der zu übertragenden Daten.
Wartezustand	S. 15, wird erreicht bei versuchtem Öffnen einer exklusiv geöffneten Datei.
Zugriffspfade	S. 15, erlauben den direkten und indirekten Zugriff zu Sätzen einer DFMS Datei.
Zugriffsverfahren	S. 13, legen die Art und Weise fest, in der eine Indexdatei reagiert (s.a. S. 14).

dietz 621

System-PASCAL

Erweiterung DFMS
Data File Management System
Beispielsammlung

DIETZ Computer SYSTEME

DIETZ Computer-Systeme · Heinrich Dietz · Solinger Straße 9 · 4330 Mülheim a. d. Ruhr · Telefon (0208) 4434-1

2-8107-01-197 Schutzgebühr DM 10,00

DIETZ Computer SYSTEME

1. Beispielsammlung

1.1. Allgemeines

Die folgenden Beispiele bauen aufeinander auf. Das heißt, die hinteren Beispiele sind nur dann sinnvoll verwendbar, wenn die ersten Beispiele eingegeben, übersetzt und ausgeführt worden sind.

Die Beispiele sind neben der Darstellung auch als anwendbare Übungen geschrieben. Durch den Dialog, den die Beispiele erzeugen, werden die Texte der Benutzeranleitung vertieft.

1.2. Alphabetische Liste

2. Beispiel PDFS01 Erzeugung einer Satz- und einer Indexdatei

```
PASCAL-01.07          DATE:  04-21-81          TIME:  10:31:20
FILE: PDFS01   PROLOG: XOSPRO   OPTIONS: PTRON, OVERFLOWON, SUBRANGEON

    1   PROGRAM PDFS01 (INPUT,OUTPUT); (*  *** DFMS-PASCAL ***  *)
    2
    3   (*  *** ERZEUGEN EINER SATZ- UND EINER INDEXDATEI ***  *)
    4
    5   (*************************************************************)
    6
    7   TYPE STRING6=ARRAY [1..6] OF CHAR;
    8        STRING15=ARRAY [1..15] OF CHAR;
    9        ANSCHRIFT=RECORD NAME : STRING15;
   10                         VORNAME : STRING15;
   11                         STRASSE : STRING15;
   12                         PLZ : 1000..8999;
   13                         ORT : STRING15
   14                    END;
   15   VAR UNIT,ANZS,ANZIN,INDEXTYP : INTEGER;
   16       DATEINAME,INDEX : STRING6;
   17       SATZ : ANSCHRIFT;
   18
   19   (*************************************************************)
   20
   21   INITPROCEDURE; BEGIN UNIT:=1 END;
   22
   23   (*************************************************************)
   24
   25   PROCEDURE DISPLAYTEXT;
   26   BEGIN
   27   WRITELN('DATEI ',DATEINAME,' MIT FOLGENDEN ATTRIBUTEN AUF');
   28   WRITELN('LAUFWERK ',UNIT:2,' ANGELEGT');
   29   IF DATEINAME()'INDDAT' THEN WRITELN('ANZAHL SAETZE = ',ANZS:3)
   30    ELSE WRITELN('ANZAHL INDIZES = ',ANZIN:4,' TYP = ',INDEXTYP:4);
   31   END;
   32
```

```
33   (*************************************************************)
34   (*                    ANWEISUNGS-TEIL                        *)
35   (*************************************************************)
36
37   BEGIN
38   DATEINAME:='SDAT01'; ANZS:=10;
39
40   CREATE (UNIT,DATEINAME,ANZS,SATZ);
41   IF DFMSERROR()0
42      THEN WRITELN('FEHLER ',DFMSERROR:4,' BEI CREATE')
43      ELSE DISPLAYTEXT;
44
45   DATEINAME:='SDAT02'; ANZS:=10;
46
47   CREATE (UNIT,DATEINAME,ANZS,SATZ);
48   IF DFMSERROR()0
49      THEN WRITELN('FEHLER ',DFMSERROR:4,' BEI CREATE')
50      ELSE DISPLAYTEXT;
51
52   DATEINAME:='INDDAT'; ANZIN:=12; INDEXTYP:=0;
53   (* SORTIERT, DOPPELTE ZUGELASSEN *)
54
55   CRIND (UNIT,DATEINAME,ANZIN,INDEX,INDEXTYP);
56   IF DFMSERROR()0
57      THEN WRITELN('FEHLER ',DFMSERROR:4,' BEI CRIND')
58      ELSE DISPLAYTEXT;
59   END.

*** WARNINGS ***

NON-STANDARD KEYWORD "INITPROCEDURE" USED
```

3. Beispiel PDFS02 Absolutzugriff über Satznummer bzw. Index

```
PASCAL-01.07          DATE:  04-29-81          TIME:  14:28:57
FILE: PDFS02   PROLOG: XOSPRO   OPTIONS: PTRON, OVERFLOWON, SUBRANGEON

    1   PROGRAM PDFS02 (INPUT,OUTPUT); (*  *** DFMS-PASCAL ***  *)
    2
    3   (*  *** ABSOLUTZUGRIFF UEBER SATZNUMMER UND INDEX ***  *)
    4
    5   (*************************************************************)
    6
    7   TYPE STRING6=ARRAY [1..6] OF CHAR;
    8        STRING15=ARRAY [1..15] OF CHAR;
    9        ANSCHRIFT=RECORD NAME : STRING15;
   10                         VORNAME : STRING15;
   11                         STRASSE : STRING15;
   12                         PLZ : 1000..8999;
   13                         ORT : STRING15
   14                    END;
   15   VAR UNIT,ARB1,ARB2,SNR : INTEGER;
   16       ZAEHLER : 1..15;
   17       SATZDATEI1,SATZDATEI2,INDEXDATEI,INDEX : STRING6;
   18       SATZ : ANSCHRIFT;
   19       ENDE : BOOLEAN;
   20
   21   (*************************************************************)
   22
   23   INITPROCEDURE;
   24   BEGIN
   25   ENDE:=FALSE;
   26   UNIT:=1;
   27   SATZDATEI1:='SDAT01';
   28   SATZDATEI2:='SDAT02';
   29   INDEXDATEI:='INDDAT';
   30   END;
   31
   32   (*************************************************************)
   33
   34   PROCEDURE FEHLER(TEXT:STRING15);
   35   BEGIN
   36   WRITELN('FEHLER : ',DFMSERROR:4,' BEI ',TEXT)
   37   END;
   38   (*************************************************************)
   39
```

```
40   PROCEDURE MENUEAUSGABE;
41   BEGIN
42   WRITELN('  SCHREIBZUGRIFF UEBER SATZNUMMER   = 0');
43   WRITELN('  SCHREIBZUGRIFF UEBER INDEX        = 1');
44   WRITELN('  LESEZUGRIFF UEBER SATZNUMMER      = 2');
45   WRITELN('  LESEZUGRIFF UEBER INDEX           = 3');
46   WRITELN('  ENDE                              = 4');
47   END;
48
49   (*****************************************************)
50
51   PROCEDURE EINGABEDATENSATZ(VAR SATZ:ANSCHRIFT);
52
53   VAR ZAEHLER : 1..15;
54
55   BEGIN
56   SATZ.NAME:='                ';
57   SATZ.VORNAME:='                ';
58   SATZ.STRASSE:='                ';
59   SATZ.ORT:='                ';
60   SATZ.PLZ:=8999;
61        WRITE('     NAME : ');
62        READLN(SATZ.NAME);
63        WRITE(' VORNAME : ');
64        READLN(SATZ.VORNAME);
65        WRITE(' STRASSE : ');
66        READLN(SATZ.STRASSE);
67        WRITE('      PLZ : ');
68        READLN(SATZ.PLZ);
69        WRITE('      ORT : ');
70        READLN(SATZ.ORT)
71   END;
72
73   (*****************************************************)
74
75   PROCEDURE DISPLAY(VAR SATZ : ANSCHRIFT);
76   BEGIN
77   WITH SATZ DO
78   WRITELN(NAME,' ',VORNAME,' ',STRASSE,' ',PLZ:5,' ',ORT)
79   END;
80
81   (*****************************************************)
82
83   FUNCTION AUSWAHL : CHAR;
84   VAR HILF: CHAR;
85   BEGIN
86   REPEAT
87      WRITE('  BITTE AUSWAHL 0 - 4 TREFFEN ');
88      READLN(HILF);
89   UNTIL (HILF>='0') AND (HILF<='4');
90   AUSWAHL:=HILF
91   END;
92
```

```
 93   (*****************************************************)
 94   (*              ANWEISUNGSTEIL                       *)
 95   (*****************************************************)
 96
 97   BEGIN
 98
 99   OPENDIRECT(UNIT,SATZDATEI1,ARB1);
100   IF DFMSERROR(>0 THEN FEHLER('OPENDIRECT     ') ELSE
101   BEGIN
102   OPENINDEXED(UNIT,SATZDATEI2,UNIT,INDEXDATEI,ARB2);
103   IF DFMSERROR(>0 THEN FEHLER('OPENINDEXED    ') ELSE
104   BEGIN
105
106   REPEAT
107   MENUEAUSGABE;
108   WRITELN;
109   WRITELN;
110     CASE AUSWAHL OF
111       '0' : BEGIN WRITE('  EINGABE SATZNUMMER : ');
112                 READLN(SNR);
113                 WRITELN;
114                 EINGABEDATENSATZ(SATZ);
115                 SELDIRECT(ARB1,SNR);
116                 IF DFMSERROR(>0 THEN FEHLER('SELDIRECT      ')
117                 ELSE BEGIN
118                 WRITES(ARB1,SATZ);
119                 IF DFMSERROR(>0 THEN FEHLER('WRITES         ')
120                 END
121             END;
122       '1' : BEGIN WRITELN('  INDEX WIRD VON DEN ERSTEN SECHS ');
123                 WRITELN('  ZEICHEN DES NAMENS ABGENOMMEN');
124                 WRITELN;
125                 WRITELN;
126                 EINGABEDATENSATZ(SATZ);
127                 FOR ZAEHLER:= 1 TO 6 DO
128                     INDEX[ZAEHLER]:=SATZ.NAME[ZAEHLER];
129                 ENTERKEY(ARB2,INDEX);
130                 IF DFMSERROR(>0 THEN FEHLER('ENTERKEY       ')
131                 ELSE BEGIN
132                 WRITES(ARB2,SATZ);
133                 IF DFMSERROR(>0 THEN FEHLER('WRITES/ENTERKEY')
134                 END
135             END;
```

```
136         '2' : BEGIN WRITE('  EINGABE SATZNUMMER : ');
137                   READLN(SNR),
138                   WRITELN;
139                   SELDIRECT(ARB1,SNR);
140                   IF DFMSERROR()0 THEN FEHLER('SELDIRECT       ')
141                   ELSE BEGIN
142                   READS(ARB1,SATZ);
143                   IF DFMSERROR()0 THEN FEHLER('READS/SELDIRECT')
144                                   ELSE DISPLAY(SATZ)
145                   END
146               END;
147         '3' : BEGIN WRITE('  EINGABE INDEX : ');
148                   READLN(INDEX);
149                   WRITELN;
150                   SELINDEXED(ARB2,INDEX);
151                   IF DFMSERROR()0 THEN FEHLER('SELINDEXED      ')
152                   ELSE BEGIN
153                   READS(ARB2,SATZ);
154                   IF DFMSERROR()0 THEN FEHLER('READ/SELINDEXED')
155                                   ELSE DISPLAY(SATZ)
156                   END
157               END;
158         '4' : ENDE:=TRUE;
159       OTHERS :
160       END
161     UNTIL ENDE
162     END
163     END
164     END
```

```
*** WARNINGS ***

NON-STANDARD OTHERS-ALTERNATIVE IN CASE-STATEMENT
NON-STANDARD KEYWORD "INITPROCEDURE" USED
```

4. Beispiel PDFS03 Sequentieller Zugriff über Satz-/Indexzeiger

```
PASCAL-01.07          DATE: 05-07-81          TIME: 08:11:12
FILE: PDFS03   PROLOG: XOSPRO   OPTIONS: PTRON, OVERFLOWON, SUBRANGEON

    1   PROGRAM PDFS03 (INPUT,OUTPUT); (*  *** DFMS-PASCAL ***  *)
    2
    3   (*  *** SEQUENTIELLER ZUGRIFF UEBER SATZ-/INDEXZEIGER ***  *)
    4
    5   (*************************************************************)
    6
    7   TYPE STRING6=ARRAY [1..6] OF CHAR;
    8        STRING15=ARRAY [1..15] OF CHAR;
    9        ANSCHRIFT=RECORD NAME : STRING15;
   10                         VORNAME : STRING15;
   11                         STRASSE : STRING15;
   12                         PLZ : 1000..8999;
   13                         ORT : STRING15
   14                    END;
   15   VAR UNIT,ARB1,ARB2,ARB3,SNR : INTEGER;
   16       SATZDATEI1,SATZDATEI2,INDEXDATEI : STRING6;
   17       SATZ : ANSCHRIFT;
   18       ENDE : BOOLEAN;
   19       DUMMY : CHAR;
   20
   21   (*************************************************************)
   22
   23   INITPROCEDURE;
   24   BEGIN
   25   ENDE:=FALSE;
   26   UNIT:=1;
   27   SATZDATEI1:='SDAT01';
   28   SATZDATEI2:='SDAT02';
   29   INDEXDATEI:='INDDAT';
   30   END;
   31
   32   (*****************************************************)
   33
   34   PROCEDURE FEHLER(TEXT:STRING15);
   35   BEGIN
   36   WRITELN('FEHLER : ',DFMSERROR:4,' BEI ',TEXT)
   37   END;
   38   (*****************************************************)
   39
   40   PROCEDURE DISPLAY(VAR SATZ : ANSCHRIFT);
   41   BEGIN
   42   WITH SATZ DO
   43   WRITELN(NAME,' ',VORNAME,' ',STRASSE,' ',PLZ:5,' ',ORT)
   44   END;
   45
```

```
46  (*****************************************************)
47  (*                 ANWEISUNGSTEIL                    *)
48  (*****************************************************)
49
50  BEGIN
51
52  OPENDIRECT(UNIT,SATZDATEI1,ARB1);
53  IF DFMSERROR()0 THEN FEHLER('OPENDIRECT     ') ELSE
54
55  BEGIN
56  OPENDIRECT(UNIT,SATZDATEI2,ARB2);
57  IF DFMSERROR()0 THEN FEHLER('OPENDIRECT     ') ELSE
58
59  BEGIN
60  OPENINDEXED(UNIT,SATZDATEI2,UNIT,INDEXDATEI,ARB3);
61  IF DFMSERROR()0 THEN FEHLER('OPENINDEXED    ') ELSE
62
63  BEGIN
64
65  WRITELN;
66  WRITELN(' SATZDATEI SDAT01 PHYSISCH SEQUENTIELL ');
67  SNR:=0;
68  SELDIRECT(ARB1,SNR);
69  WHILE DFMSERROR()100 DO
70  BEGIN
71  READNEXT(ARB1,SATZ);
72  CASE DFMSERROR OF
73       0:   DISPLAY(SATZ);
74     100:   WRITELN(' ENDE DER DATEI ');
75     101:   BEGIN
76            WRITELN(' SATZ IST LEER ');
77            NEXT(ARB1)
78            END;
79  OTHERS:   FEHLER('SATZ-SEQUENT.  ')
80  END
81  END;
82
83  WRITELN;
84  WRITELN(' NAECHSTES BILD, BITTE (CR) ZWEIMAL DRUECKEN');
85  READLN(DUMMY);
86
87  WRITELN;
88  WRITELN(' SATZDATEI SDAT02 PHYSISCH SEQUENTIELL ');
89  SNR:=0;
90  SELDIRECT(ARB2,SNR);
91  WHILE DFMSERROR()100 DO
92  BEGIN
93  READNEXT(ARB2,SATZ);
```

```
 94   CASE DFMSERROR OF
 95        0:   DISPLAY(SATZ);
 96      100:   WRITELN(' ENDE DER DATEI ');
 97      101:   BEGIN
 98             WRITELN(' SATZ IST LEER ');
 99             NEXT(ARB2)
100             END;
101   OTHERS:   FEHLER('SATZ-SEQUENT.  ')
102   END
103   END;
104
105   WRITELN;
106   WRITELN(' NAECHSTES BILD, BITTE (CR) ZWEIMAL DRUECKEN');
107   READLN(DUMMY);
108
109   WRITELN;
110   WRITELN(' SATZDATEI SDAT02 LOGISCH SEQUENTIELL ');
111   FIRST(ARB3);
112   WHILE DFMSERROR()100 DO
113   BEGIN
114   READNEXT(ARB3,SATZ);
115   CASE DFMSERROR OF
116        0:   DISPLAY(SATZ);
117      100:   WRITELN(' ENDE DER DATEI');
118   OTHERS:   FEHLER('LOGISCH SEQU.  ')
119   END
120   END
121   END
122   END
123   END
124   END.

*** WARNINGS ***

NON-STANDARD OTHERS-ALTERNATIVE IN CASE-STATEMENT
NON-STANDARD KEYWORD "INITPROCEDURE" USED
```

5. Beispiel PDFS04 Suchen eines Satzes über einen Index

```
PASCAL-01.07          DATE: 05-11-81          TIME: 19:28:54
FILE: PDFS04   PROLOG: XOSPRO   OPTIONS: PTRON, OVERFLOWON, SUBRANGEON

  1  PROGRAM PDFS04 (INPUT,OUTPUT); (*  *** DFMS-PASCAL ***  *)
  2
  3  (*  *** SUCHEN EINES SATZES UEBER EINEN INDEX         ***
  4
  5  (*************************************************************)
  6
  7  TYPE STRING6=ARRAY [1..6] OF CHAR;
  8       STRING15=ARRAY [1..15] OF CHAR;
  9       ANSCHRIFT=RECORD NAME : STRING15;
 10                        VORNAME : STRING15;
 11                        STRASSE : STRING15;
 12                        PLZ : 1000..8999;
 13                        ORT : STRING15
 14                   END;
 15
 16  VAR SATZ : ANSCHRIFT;
 17      MENGEAUSWAHL : SET OF CHAR;
 18      LT,GT,EQ,LE,GE : CHAR;
 19      SUCH,INDEX,SATZDATEI,INDEXDATEI : STRING6;
 20      ARB1,UNIT : INTEGER;
 21      ZAEHLER : 1..6;
 22      ENDE : BOOLEAN;
 23
 24  (*************************************************************)
 25
 26  INITPROCEDURE;
 27  BEGIN
 28  ENDE:=FALSE;
 29  UNIT:=1;
 30  SATZDATEI:='SDAT02';
 31  INDEXDATEI:='INDDAT';
 32  MENGEAUSWAHL:=['(',')','=','L','G','E'];
 33  LT:='('; GT:=')'; EQ:='='; LE:='L'; GE:='G';
 34  END;
 35
 36  (*************************************************)
 37
 38  PROCEDURE FEHLER(TEXT:STRING15);
 39  BEGIN
 40  WRITELN('FEHLER : ',DFMSERROR:4,' BEI ',TEXT)
 41  END;
 42  (*************************************************)
```

```
43
44   PROCEDURE EINGABE(VAR INDEX:STRING6);
45   BEGIN
46   WRITELN;
47   WRITELN;
48   WRITELN(' BITTE GEBEN SIE NUN DEN INDEX MIT SECHS STELLEN ');
49   WRITELN(' EIN  DABEI IST ES AUCH MOEGLICH, STERNCHEN ''*'' ');
50   WRITELN(' FUER NICHT RELEVANTE STELLEN EINZUGEBEN.');
51   WRITELN(' Z.B.   DIE***');
52   WRITELN;
53   READLN(INDEX)
54   END;
55
56   (*******************************************************)
57
58   PROCEDURE DISPLAY(VAR SATZ:ANSCHRIFT);
59   BEGIN
60   WITH SATZ DO
61   WRITELN(NAME,' ',VORNAME,' ',STRASSE,' ',PLZ:5,' ',ORT)
62   END;
63
64   (*******************************************************)
65
66   PROCEDURE DISPLAYAUSWAHL;
67   BEGIN
68          WRITELN(' FOLGENDE MOEGLICHKEITEN SIND GEGEBEN  ');
69          WRITELN(' <  : KLEINER');
70          WRITELN(' >  : GROESSER');
71          WRITELN(' =  : GLEICH');
72          WRITELN(' L  : KLEINER ODER GLEICH');
73          WRITELN(' G  : GROESSER ODER GLEICH');
74          WRITELN(' E  : ENDE');
75   END;
76
77   (*******************************************************)
78
79   FUNCTION AUSWAHL : CHAR;
80   VAR HILF: CHAR;
81   BEGIN
82   REPEAT WRITELN(' BITTE AUSWAHL TREFFEN ');
83          WRITELN(' <  >  =  L  G  E ');
84          READLN(HILF)
85   UNTIL HILF IN MENGEAUSWAHL;
86   AUSWAHL:=HILF
87   END;
```

```
 88
 89  (*****************************************************)
 90  (*                  HAUPTPROGRAMM                    *)
 91  (*****************************************************)
 92
 93
 94  BEGIN
 95  OPENINDEXED(UNIT,SATZDATEI,UNIT,INDEXDATEI,ARB1);
 96  IF DFMSERROR()0 THEN FEHLER('OPENINDEXED    ') ELSE
 97  BEGIN
 98  REPEAT EINGABE (INDEX);
 99         DISPLAYAUSWAHL;
100         CASE AUSWAHL OF
101              '<' : SEKEY(ARB1,INDEX,LT,SUCH);
102              '>' : SEKEY(ARB1,INDEX,GT,SUCH);
103              '=' : SEKEY(ARB1,INDEX,EQ,SUCH);
104              'L' : SEKEY(ARB1,INDEX,LE,SUCH);
105              'G' : SEKEY(ARB1,INDEX,GE,SUCH);
106          OTHERS : ENDE:=TRUE
107         END;
108         IF DFMSERROR()0 THEN FEHLER('SEKEY          ')
109            ELSE
110            IF NOT ENDE THEN BEGIN
111                             READS(ARB1,SATZ);
112                             DISPLAY(SATZ)
113                             END
114  UNTIL ENDE
115  END
116  END.
```

```
*** WARNINGS ***

NON-STANDARD OTHERS-ALTERNATIVE IN CASE-STATEMENT
NON-STANDARD KEYWORD "INITPROCEDURE" USED
```

6. Beispiel PDFS05 Blättern über Index

```
PASCAL-01.07          DATE: 05-19-81          TIME: 08:36:04
FILE: PDFS05   PROLOG: XOSPRO   OPTIONS: PTRON, OVERFLOWON, SUBRANGEON

    1   PROGRAM PDFS05 (INPUT,OUTPUT); (*  *** DFMS-PASCAL ***  *)
    2
    3   (*  *** BLAETTERN UEBER INDEX                          ***)
    4
    5   (*************************************************************)
    6
    7   TYPE STRING6=ARRAY [1..6] OF CHAR;
    8        STRING15=ARRAY [1..15] OF CHAR;
    9        ANSCHRIFT=RECORD NAME : STRING15;
   10                         VORNAME : STRING15;
   11                         STRASSE : STRING15;
   12                         PLZ : 1000..8999;
   13                         ORT : STRING15
   14                    END;
   15
   16   VAR SATZ : ANSCHRIFT;
   17       MENGEAUSWAHL : SET OF CHAR;
   18       ANTW,LT,GT,EQ,LE,GE : CHAR;
   19       SUCH,INDEX,SATZDATEI,INDEXDATEI : STRING6;
   20       ARB1,UNIT : INTEGER;
   21       ZAEHLER : 1..6;
   22       ENDE : BOOLEAN;
   23
   24   (*************************************************************)
   25
   26   INITPROCEDURE;
   27   BEGIN
   28   ENDE:=FALSE;
   29   UNIT:=1;
   30   SATZDATEI:='SDAT02';
   31   INDEXDATEI:='INDDAT';
   32   MENGEAUSWAHL:=['(',')','=','L','G','E'];
   33   LT:='(', GT:=')'; EQ:='='; LE:='L'; GE:='G';
   34   END;
   35
   36   (*************************************************)
```

```
   37
   38   PROCEDURE DISPLAY(VAR SATZ:ANSCHRIFT);
   39   BEGIN
   40   WITH SATZ DO
   41   WRITELN(NAME,' ',VORNAME,' ',STRASSE,' ',PLZ,' ',ORT)
   42   END;
   43
   44   (*******************************************************)
   45
   46   PROCEDURE FEHLER(TEXT:STRING15);
   47   BEGIN
   48   WRITELN('FEHLER : ',DFMSERROR:4,' BEI ',TEXT)
   49   END;
   50
   51   (*******************************************************)
   52
   53   PROCEDURE BLAETTERN;
   54   VAR ZAEHLER : 1..6;
   55       WEITERBLAETTERN : BOOLEAN;
   56       VRE : CHAR;
   57   BEGIN
   58   WEITERBLAETTERN:=TRUE;
   59   WHILE WEITERBLAETTERN DO
   60         BEGIN
   61         REPEAT (* EINGABE *)
   62      .  WRITELN;
   63         WRITELN(' VORWAERTS = V, RUECKWAERTS = R, ENDE = E');
   64         WRITELN;
   65         READLN(VRE);
   66         UNTIL (VRE='V') OR (VRE='R') OR (VRE='E');
   67         CASE VRE OF
   68              'V': NEXT(ARB1);
   69              'R': SEKEY(ARB1,INDEX,GT,SUCH);
   70              'E': WEITERBLAETTERN:=FALSE
   71         END;
******         ^
      1. POSSIBILITY OF UNDEFINED CASELABEL AT RUNTIME

   72         IF DFMSERROR()0 THEN FEHLER ('BLAETTERN      ')
   73            ELSE IF WEITERBLAETTERN THEN
   74                    BEGIN
   75                    READS(ARB1,SATZ);
   76                    DISPLAY(SATZ);
   77                    FOR ZAEHLER:=1 TO 6 DO
   78                        INDEX[ZAEHLER]:=SATZ.NAME[ZAEHLER]
   79                    END
   80         END
   81   END.
   82
   83   (*******************************************************)
   84
```

```
 85   PROCEDURE EINGABE(VAR INDEX:STRING6);
 86   BEGIN
 87   WRITELN,
 88   WRITELN;
 89   WRITELN(' BITTE GEBEN SIE NUN DEN INDEX MIT SECHS STELLEN ');
 90   WRITELN(' EIN. DABEI IST ES AUCH MOEGLICH, STERNCHEN ''*'' ');
 91   WRITELN(' FUER NICHT RELEVANTE STELLEN EINZUGEBEN.');
 92   WRITELN(' Z.B.   DIE***');
 93   WRITELN;
 94   READLN(INDEX)
 95   END;
 96
 97   (*****************************************************)
 98
 99   PROCEDURE DISPLAYAUSWAHL;
100   BEGIN
101          WRITELN(' FOLGENDE MOEGLICHKEITEN SIND GEGEBEN  ');
102          WRITELN(' < : KLEINER');
103          WRITELN(' > : GROESSER');
104          WRITELN(' = : GLEICH');
105          WRITELN(' L : KLEINER ODER GLEICH');
106          WRITELN(' G : GROESSER ODER GLEICH');
107          WRITELN(' E : ENDE');
108   END;
109
110   (*****************************************************)
111
112   FUNCTION AUSWAHL : CHAR;
113   VAR HILF: CHAR;
114   BEGIN
115   REPEAT WRITELN(' BITTE AUSWAHL TREFFEN ');
116          WRITELN('...(..)..=..L..G..E...');
117          READLN(HILF)
118   UNTIL HILF IN MENGEAUSWAHL;
119   AUSWAHL:=HILF
120   END;
121
```

```
  122   (*****************************************************)
  123   (*                    HAUPTPROGRAMM                  *)
  124   (*****************************************************)
  125
  126
  127   BEGIN
  128   OPENINDEXED(UNIT,SATZDATEI,UNIT,INDEXDATEI,ARB1);
  129   IF DFMSERROR()0 THEN FEHLER('OPENINDEXED     ') ELSE
  130   BEGIN
  131   REPEAT EINGABE (INDEX);
  132          DISPLAYAUSWAHL;
  133          CASE AUSWAHL OF
  134               '(' : SEKEY(ARB1,INDEX,LT,SUCH);
  135               ')' : SEKEY(ARB1,INDEX,GT,SUCH);
  136               '=' : SEKEY(ARB1,INDEX,EQ,SUCH);
  137               'L' : SEKEY(ARB1,INDEX,LE,SUCH);
  138               'G' : SEKEY(ARB1,INDEX,GE,SUCH);
  139               'E' : ENDE:=TRUE
  140           END;
******          ^
      1. POSSIBILITY OF UNDEFINED CASELABEL AT RUNTIME

  141           IF DFMSERROR()0 THEN FEHLER('SEKEY           ')
  142              ELSE
  143              IF NOT ENDE THEN BEGIN
  144                                READS(ARB1,SATZ);
  145                                DISPLAY(SATZ);
  146                                REPEAT (* EINGABE *)
  147                                WRITELN;
  148                                WRITE(' BLAETTERN J/N ');
  149                                READLN(ANTW);
  150                                UNTIL (ANTW='J') OR (ANTW='N');
  151                                IF ANTW='J' THEN BLAETTERN
  152                                END
  153   UNTIL ENDE
  154   END
  155   END.

*** WARNINGS ***

NON-STANDARD KEYWORD "INITPROCEDURE" USED
```

7. Beispiel PDFS06 Lesen mit Satzsperre und Modifikation

```
PASCAL-01.07          DATE: 05-25-81          TIME: 20:06:43
FILE: PDFS06  PROLOG: XOSPRO  OPTIONS: PTRON, OVERFLOWON, SUBRANGEON

    1   PROGRAM PDFS06 (INPUT,OUTPUT); (*  *** DFMS-PASCAL ***  *)
    2
    3   (*  *** LESEN MIT SATZSPERRE UND MODIFIKATION         ***)
    4
    5   (*************************************************************)
    6
    7   TYPE STRING6=ARRAY [1..6] OF CHAR;
    8        STRING15=ARRAY [1..15] OF CHAR;
    9        ANSCHRIFT=RECORD NAME : STRING15;
   10                         VORNAME : STRING15;
   11                         STRASSE : STRING15;
   12                         PLZ : 1000..8999;
   13                         ORT : STRING15
   14                    END;
   15
   16   VAR SATZ : ANSCHRIFT;
   17       MENGEAUSWAHL : SET OF CHAR;
   18       ANTWORT,LT,GT,EQ,LE,GE : CHAR;
   19       SUCH,INDEX,SATZDATEI,INDEXDATEI : STRING6;
   20       ARB1,UNIT : INTEGER;
   21       ZAEHLER : 1..6;
   22       ENDE : BOOLEAN;
   23
   24   (*************************************************************)
   25
   26   INITPROCEDURE;
   27   BEGIN
   28   ENDE:=FALSE;
   29   UNIT:=1;
   30   SATZDATEI:='SDAT02';
   31   INDEXDATEI:='INDDAT';
   32   MENGEAUSWAHL:=['<','>','=','L','G','E'];
   33   LT:='<'; GT:='>'; EQ:='='; LE:='L'; GE:='G';
   34   END;
   35
   36   (*************************************************)
   37
   38   PROCEDURE DISPLAY(VAR SATZ:ANSCHRIFT);
   39   BEGIN
   40   WITH SATZ DO
   41   WRITELN(NAME,' ',VORNAME,' ',STRASSE,' ',PLZ:5,' ',ORT)
   42   END;
   43
   44   (*************************************************)
   45
```

```
46  PROCEDURE FEHLER(TEXT:STRING15);
47  BEGIN
48  WRITELN('FEHLER : ',DFMSERROR:4,' BEI ',TEXT)
49  END;
50
51  (*****************************************************)
52
53  PROCEDURE AENDERN;
54  VAR ZAEHLER : 1..15;
55      NAMEX,VORNAMEX,STRASSEX,ORTX:STRING15;
56      PLZX : 1000..8999;
57      ANTWORT,AKTUELLERINDEX : CHAR;
58      NEUERINDEX : BOOLEAN;
59
60  BEGIN
61  NAMEX:='                ';
62  VORNAMEX:='                ';
63  STRASSEX:='                ';
64  ORTX:='                ';
65  WITH SATZ DO BEGIN
66  REPEAT
67    WRITELN(' ****** ALTEN WERT UEBERNEHMEN MIT ^ (CR) ******');
68    WRITELN;
69    WRITELN;
70    WRITE(' NAME ALT       : ',NAME,' NAME NEU     : ');
71    READLN(NAMEX);
72    IF NAMEX[1]<>'^' THEN BEGIN NAME:=NAMEX; NEUERINDEX:=TRUE END;
73    WRITELN;
74    WRITE(' VORNAME ALT    : ',VORNAME,' VORNAME NEU  : ');
75    READLN(VORNAMEX);
76    IF VORNAMEX[1]<>'^' THEN VORNAME:=VORNAMEX;
77    WRITELN;
78    WRITE(' STRASSE ALT    : ',STRASSE,' STRASSE NEU  : ');
79    READLN(STRASSEX);
80    IF STRASSEX[1]<>'^' THEN STRASSE:=STRASSEX;
81    WRITELN;
82    WRITELN(' ****** POSTLEITZAHL MUSS EINGEGEBEN WERDEN ******');
83    WRITELN;
84    WRITE(' PLZ  ALT       : ',PLZ:5, '            PLZ  NEU     : ');
85    READLN(PLZ);
86    WRITELN;
87    WRITE(' ORT  ALT       : ',ORT,' ORT  NEU      : ');
88    READLN(ORTX);
89    IF ORTX[1]<>'^' THEN ORT:=ORTX;
```

```
 90      WRITELN;
 91      WRITELN(' ****** NEUER DATENSATZ ******');
 92      WRITELN;
 93      DISPLAY(SATZ);
 94      REPEAT (* EINGABE *)
 95      WRITELN;
 96      WRITE(' SIND DIE AENDERUNGEN OK ? J/N ');
 97      READLN(ANTWORT);
 98      UNTIL (ANTWORT='J') OR (ANTWORT='N')
 99      UNTIL (ANTWORT='J');
100       FOR ZAEHLER:= 1 TO 6 DO INDEX[ZAEHLER]:=SATZ.NAME[ZAEHLER];
101       AKTUELLERINDEX:=CHR(0);
102       MODIFY(ARB1,SATZ);
103       IF NEUERINDEX THEN RENAMEKEY(ARB1,AKTUELLERINDEX,INDEX)
104                     ELSE NEXT (ARB1);
105       END
106       END;
107
108   (*******************************************************)
109
110   PROCEDURE BLAETTERN;
111   VAR ZAEHLER : 1..6;
112       WEITERBLAETTERN : BOOLEAN;
113       VRE : CHAR;
114   BEGIN
115   WEITERBLAETTERN:=TRUE;
116   WHILE WEITERBLAETTERN DO
117         BEGIN
118         REPEAT (* EINGABE *)
119         WRITELN;
120         WRITELN(' VORWAERTS = V, RUECKWAERTS = R, ENDE = E');
121         WRITELN;
122         READLN(VRE);
123         UNTIL (VRE='V') OR (VRE='R') OR (VRE='E');
124         CASE VRE OF
125              'V': NEXT(ARB1);
126              'R': SEKEY(ARB1,INDEX,GT,SUCH);
127              'E': WEITERBLAETTERN:=FALSE
128         END;
******        ^
      1. POSSIBILITY OF UNDEFINED CASELABEL AT RUNTIME

129         IF DFMSERROR()0 THEN FEHLER ('BLAETTERN      ')
130            ELSE IF WEITERBLAETTERN THEN
131                    BEGIN
132                    READS(ARB1,SATZ);
133                    DISPLAY(SATZ);
```

```
134                     FOR ZAEHLER:=1 TO 6 DO
135                         INDEX[ZAEHLER]:=SATZ.NAME[ZAEHLER]
136                     END
137         END
138   END.
139
140   (*****************************************************)
141
142   PROCEDURE EINGABE(VAR INDEX:STRING6);
143   BEGIN
144   WRITELN;
145   WRITELN;
146   WRITELN(' BITTE GEBEN SIE NUN DEN INDEX MIT SECHS STELLEN ');
147   WRITELN(' EIN. DABEI IST ES AUCH MOEGLICH, STERNCHEN ''*'' ');
148   WRITELN(' FUER NICHT RELEVANTE STELLEN EINZUGEBEN.');
149   WRITELN(' Z.B.   DIE***');
150   WRITELN;
151   READLN(INDEX)
152   END;
153
154   (*****************************************************)
155
156   PROCEDURE DISPLAYAUSWAHL;
157   BEGIN
158         WRITELN(' FOLGENDE MOEGLICHKEITEN SIND GEGEBEN  ');
159         WRITELN(' < : KLEINER');
160         WRITELN(' > : GROESSER');
161         WRITELN(' = : GLEICH');
162         WRITELN(' L : KLEINER ODER GLEICH');
163         WRITELN(' G : GROESSER ODER GLEICH');
164         WRITELN(' E : ENDE');
165   END;
166
167   (*****************************************************)
168
```

```
  169   FUNCTION AUSWAHL : CHAR;
  170   VAR HILF: CHAR;
  171   BEGIN
  172   REPEAT WRITELN(' BITTE AUSWAHL TREFFEN ');
  173          WRITELN('...<..)..=..L..G..E...');
  174          READLN(HILF)
  175   UNTIL HILF IN MENGEAUSWAHL;
  176   AUSWAHL:=HILF
  177   END;
  178
  179   (*****************************************************)
  180   (*                   HAUPTPROGRAMM                   *)
  181   (*****************************************************)
  182
  183
  184   BEGIN
  185   OPENINDEXED(UNIT,SATZDATEI,UNIT,INDEXDATEI,ARB1);
  186   IF DFMSERROR()0 THEN FEHLER('OPENINDEXED    ') ELSE
  187   BEGIN
  188   REPEAT EINGABE (INDEX);
  189          DISPLAYAUSWAHL;
  190          CASE AUSWAHL OF
  191               '<' : SEKEY(ARB1,INDEX,LT,SUCH);
  192               '>' : SEKEY(ARB1,INDEX,GT,SUCH);
  193               '=' : SEKEY(ARB1,INDEX,EQ,SUCH);
  194               'L' : SEKEY(ARB1,INDEX,LE,SUCH);
  195               'G' : SEKEY(ARB1,INDEX,GE,SUCH);
  196               'E' : ENDE:=TRUE
  197           END;
******          ^
      1. POSSIBILITY OF UNDEFINED CASELABEL AT RUNTIME

  198           IF DFMSERROR()0 THEN FEHLER('SEKEY          ')
  199              ELSE
  200              IF NOT ENDE THEN BEGIN
  201                                UPDATE(ARB1,SATZ);
  202                                DISPLAY(SATZ);
```

```
203                                 REPEAT (* EINGABE *)
204                                 WRITELN;
205                                 WRITE(' BLAETTERN J/N ');
206                                 READLN(ANTWORT);
207                                 UNTIL (ANTWORT='J') OR (ANTWORT='N');
208                                 IF ANTWORT='J' THEN BLAETTERN;
209                                 REPEAT (* EINGABE *)
210                                 WRITELN;
211                                 WRITE(' AENDERN   J/N ');
212                                 READLN(ANTWORT);
213                                 UNTIL (ANTWORT='J') OR (ANTWORT='N');
214                                 IF ANTWORT='J' THEN AENDERN
215                                 END
216   UNTIL ENDE
217   END
218   END.

*** WARNINGS ***

NON-STANDARD KEYWORD "INITPROCEDURE" USED
```

8. Beispiel PDFS07 Nachträgliches Invertieren einer Satzdatei

```
PASCAL-01.07          DATE: 05-26-81          TIME: 13:11:11
FILE: PDFS07   PROLOG: XOSPRO   OPTIONS: PTRON, OVERFLOWON, SUBRANGEON

    1   PROGRAM PDFS07 (INPUT,OUTPUT); (*  *** DFMS-PASCAL ***  *)
    2
    3   (*  *** NACHTRAEGLICHES INVERTIEREN EINER SATZ-DATEI  ***  *)
    4
    5   (*************************************************************)
    6
    7   TYPE STRING6=ARRAY [1..6] OF CHAR;
    8        STRING15=ARRAY [1..15] OF CHAR;
    9        ANSCHRIFT=RECORD NAME : STRING15;
   10                         VORNAME : STRING15;
   11                         STRASSE : STRING15;
   12                         PLZ : 1000..8999;
   13                         ORT : STRING15
   14                    END;
   15
   16   VAR UNIT,ARB1 : INTEGER;
   17       SATZDATEI,INDEXDATEI : STRING6;
   18       SATZ : ANSCHRIFT;
   19       ANZAHLINDIZES,TYP : INTEGER;
   20       INDEX : STRING15;
   21
   22   (*************************************************************)
   23
   24   INITPROCEDURE;
   25   BEGIN
   26   UNIT:=1;
   27   SATZDATEI:='SDAT02'
   28   END;
   29
   30   (*************************************************************)
   31
   32   PROCEDURE FEHLER(TEXT:STRING15);
   33   BEGIN
   34   WRITELN('FEHLER : ',DFMSERROR:4,' BEI ',TEXT)
   35   END;
   36   (*************************************************************)
   37
```

```
38   PROCEDURE DISPLAY(VAR SATZ:ANSCHRIFT);
39   BEGIN
40   WITH SATZ DO
41   WRITELN(NAME,' ',VORNAME,' ',STRASSE,' ',PLZ:5,' ',ORT)
42   END;
43
44   (*****************************************************)
45
46   PROCEDURE DRUCKAUSGABE;
47   BEGIN
48   WHILE DFMSERROR<>100 DO
49     BEGIN
50     READNEXT(ARB1,SATZ);
51     IF DFMSERROR=0 THEN DISPLAY(SATZ)
52     END;
53   WRITELN;WRITELN('ENDE DER DATEI ERREICHT  *** EOF ***')
54   END;
55
56   (*****************************************************)
57   (*              ANWEISUNGSTEIL                       *)
58   (*****************************************************)
59     *
60   BEGIN
61
62   WRITELN;
63   WRITELN(' DATEI NACH NAMEN SORTIERT ');
64   WRITELN;
65   INDEXDATEI:='INDDAT';
66   OPENINDEXED(UNIT,SATZDATEI,UNIT,INDEXDATEI,ARB1);
67   IF DFMSERROR<>0 THEN FEHLER('OPENINDEXED 1  ')
68                   ELSE BEGIN
69                        DRUCKAUSGABE;
70                        CLOSE(ARB1)
71                        END;
72
```

```
 73   WRITELN;
 74   WRITELN(' DATEI NACH ORTEN SORTIERT');
 75   WRITELN;
 76   INDEXDATEI:='INVERT';
 77   ANZAHLINDIZES:=12; TYP:=0;
 78   KILL(UNIT,INDEXDATEI);
 79   IF DFMSERROR=65 THEN FEHLER('KILL           ');
 80   CRIND(UNIT,INDEXDATEI,ANZAHLINDIZES,INDEX,TYP);
 81   IF DFMSERROR()0 THEN FEHLER('CRIND          ')
 82   ELSE BEGIN
 83   OPENDIRECT(UNIT,INDEXDATEI,ARB1);
 84   IF DFMSERROR()0 THEN FEHLER('OPENDIRECT     ')
 85   ELSE BEGIN
 86   KEYINVERT(UNIT,SATZDATEI,SATZ,SATZ.ORT,ARB1);
 87   IF DFMSERROR()0 THEN FEHLER('KEYINVERT      ')
 88   ELSE CLOSE (ARB1)
 89   END
 90   END;
 91   OPENINDEXED(UNIT,SATZDATEI,UNIT,INDEXDATEI,ARB1);
 92   IF DFMSERROR()0 THEN FEHLER('OPENINDEXED 2  ')
 93                   ELSE BEGIN
 94                        DRUCKAUSGABE;
 95                        CLOSE(ARB1)
 96                        END;
 97   END.
```

*** WARNINGS ***

NON-STANDARD KEYWORD "INITPROCEDURE" USED

dietz 621

System-PASCAL

Erweiterung DFMS
Data File Management System
Benutzeranleitung

DIETZ Computer SYSTEME

DIETZ Computer-Systeme · Heinrich Dietz · Solinger Straße 9 · 4330 Mülheim a. d. Ruhr · Telefon (0208) 4434-1

2-8107-01-196 Schutzgebühr DM 27,50

DIETZ Computer SYSTEME

1. Allgemeines

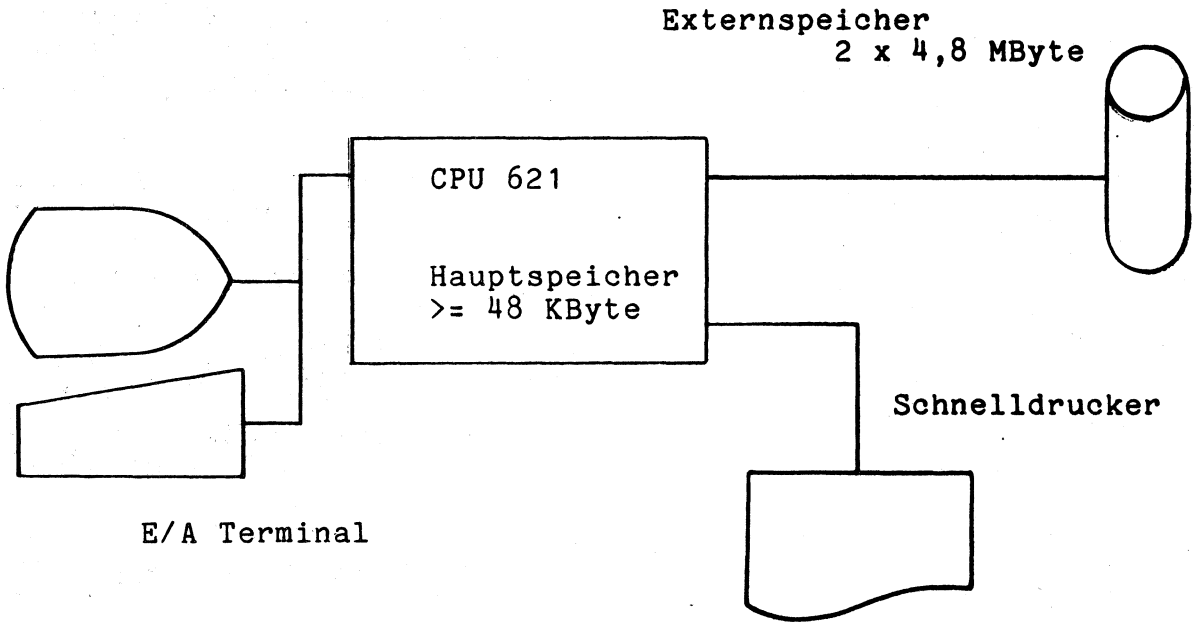
1.1. Aufgabe

Die in einem Prolog als Standardprozeduren vereinbarten DFMS-Aufrufe dienen dazu, satzstruktuierte Dateien aufzubauen, zu beschreiben, lesen und zu modifizieren.
Es kann sowohl direkt über die Satznummer als auch indirekt über einen Index auf einen Satz zugegriffen werden. Neben den Absolutzugriffen sind auch satzsequentielle und indexsequentielle Zugriffe möglich.

Die vier möglichen Zugriffswege bzw. die daraus resultierenden Organisationsformen werden durch zwei verschiedene Dateiformen erzeugt.
Vom Anwender können sowohl eine Satzdatei als auch eine Indexdatei aufgebaut werden.
Da jede Dateiform für sich allein sowohl den wahlfreien (RANDOM) als auch sequentiellen Zugriff erlaubt, ist dieses System äußerst flexibel. Erst durch ein Programm wird eine Indexdatei einer Satzdatei zugeordnet. Es können aber auch mehrere Indexdateien sein.
Die Organisationsform der indexsequentiellen Datei (ISAM) wird durch ein sogenanntes verkettetes Öffnen erreicht.

1.2. Voraussetzungen und Generierungsbeispiel

Folgendes Schaubild soll die hardwaremäßige Grundausstattung verdeutlichen. Die Software-Voraussetzungen können dem Generierungsbeispiel bzw. dem Kapitel 1.2.1 (Start des Systems) entnommen werden.



1.2.1. Generierungsbeispiel

Eine ausführliche Beschreibung der Einbindeparameter des Moduls DFMS ist in der Benutzeranleitung XOS-Modul DFMS enthalten.

Speicherbelegung und Partitionsgröße

```
SYTEMFILE (UNIT,NAME) ?1,TSOS-X (CR)

*** TSOS-EF0-06.2 ***

 DIAL
*PART (CR)
PART, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0   FREE:    184
 DIAL
*MEM (CR)
BEGIN: 'BC00   END: 'BFFF   LENGTH:   1024
 DIAL
*
```

```
 DIAL
*
SERV,MODULE,1,MDFMS,,32,512,0,9 (CR)
 MDFMS - 15.04
ARBEITSNUMMERN   32
PUFFER     0

MODULE MDFMS FROM '6000 TO '7242 LENGTH '1243
 DIAL
*LOP,400C,R,0001 (CR)
C0 00
 DIAL
*PART*,256 (CR)
 DIAL
*PART (CR)
PART,256,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0    FREE     26
 DIAL
*MEM (CR)
BEGIN: '8000     END: 'FFFF     LENGTH:  32768
 DIAL
 *CRE,1,OSCOMP,256 (CR)
 DIAL
*NEW,1,OSCOMP (CR)
 DIAL
*RUN,1,COMSYS (CR)
*** COMSYS-04.02 ***
 COMSYS
*
```

1.2.2. Start des Systems

Folgende Bedienungs- bzw. Kommandofolge soll aufzeigen, welches Systemumfeld benötigt wird. Außerdem ist zu beachten, daß genügend Freibereich auf der Systemplatte vorhanden ist, da das System COMSYS transiente Dateien aufbaut (!Cu).
Außerdem muß im Hauptspeicher der Modul MDFMS eingebunden sein (s.a. 1.2.1 Generierung). Dadurch bedingt, fängt die Partition (Abfrage mit MEM <cr>) erst bei Adresse ´8000 an.
Diese Angabe bezieht sich auf ein System DIETZ 621X2 und dem in Kapitel 1.2 enthaltenem Generierungsbeispiel.

```
*** XOSBOOT-07.01 ***
SYSTEMFILE (UNIT,NAME) ?1,OSCOMP (CR)
*** TSOS-EF0-06.02 ***
 DIAL
*MEM (CR)
BEGIN: '8000     END: 'FFFF     LENGTH  32768
 DIAL
*RUN,1,COMGEN (CR)
*** COMSYS-04.02 ***
 LANGUAGE (COB, PAS, FOR) ? PASCAL (CR)
*** ALL OPTIONS FOR USER   0 ARE DEFAULT-VALUES ***
 DIAL
*RUN,1,COMSYS (CR)
*** COMSYS-04.02 ***
 COMSYS
*
```

*1)

Die Angabe <cr> heißt, daß das Kommando "MEM" durch die Taste CR abgeschlossen wird.

Startbeispiel und überprüfung der Dateien

```
 COMSYS

*SERV,LIST,M,1,,WF** (CR)

    41  WF00        7848     256  '00  '34
    68  WF10       15640    1000  '00  '34
    69  WF20       16640     600  '00  '34

 COMSYS

*SERV,LIST,M,1,,COMGEN (CR)

     5  COMGEN      1240      50  '37  '25

 COMSYS

*SERV,LIST,M,1,,COMSYS

     3  COMSYS       368       6  '30  '01

 COMSYS

*SERV,LIST,M,1,,MCOMSY (CR)

     4  MCOMSY       376     861  '37  '25

 COMSYS

*SERV,LIST,M,1,,XOSP** (CR)

    18  XOSPRO      5476      80  '19  '00
    45  XOSPAS      8764    1197  '36  '00

 COMSYS

*SERV,LIST,M,1,,PASBIB (CR)

    48  PASBIB     10384    1528  '00  '34

 COMSYS

*SERV,LIST,M,1,,!***** (CR)

    67  !INFOV     15384     256  '00  '34
    70  !C0        17324    1041  '00

 COMSYS

*
```

1.3. Arbeiten mit dem System

Dieses Kapitel zeigt an einem Beispiel den Dialog einer übersetzung auf.
Eine vollständige Beschreibung aller Möglichkeiten des Bediensystems und des Binders ist in der Benutzeranleitung "Bediensystem und Binder" enthalten.

```
 DIAL
*RUN,1,COMSYS (CR)
*** COMSYS-04.02 ***
 COMSYS
*DEVICE,3 (CR)
 COMSYS
*SOURCE (CR)
 COMSYS
*COMPILE,1,PDFS03,1,PDFX03 (CR)
 *** PASCAL - 1.07 ***
*** NO ERRORS DETECTED ***
*** COMPILATION TERMINATED ***
 COMSYS
*BINDER (CR)
*** BINDER-03.09 ***
*ENTER,1,PDFX01 (CR)
*OBJECT,1,PDF.01 (CR)
 COMSYS
*RUN,1,PDF.01 (CR)
 DATEI SDAT01 MIT FOLGENDEN ATTRIBUTEN AUF
 LAUFWERK  1 ANGELEGT
 ANZAHL SAETZE  =   10
  COMSYS
```

```
 1  PROGRAM PDFS01 (INPUT,OUTPUT); (*  *** DFMS-PASCAL ***  *)
 2
 3  (*  *** ERZEUGEN EINER SATZ- UND EINER INDEXDATEI ***  *)
 4
 5  (*********************************************************)
 6
 7  TYPE STRING6=ARRAY [1..6] OF CHAR;
 8       STRING15=ARRAY [1..15] OF CHAR;
 9       ANSCHRIFT=RECORD NAME : STRING15;
10                        VORNAME : STRING15;
11                        STRASSE : STRING15;
12                        PLZ : 1000..8999;
13                        ORT : STRING15
14                   END;
15  VAR UNIT,ANZS,ANZIN,INDEXTYP : INTEGER;
16      DATEINAME,INDEX : STRING6;
17      SATZ : ANSCHRIFT;
18
19  (*********************************************************)
20
21  INITPROCEDURE; BEGIN UNIT:=1 END;
22
23  (*********************************************************)
24
25  PROCEDURE DISPLAYTEXT;
26  BEGIN
27  WRITELN('DATEI ',DATEINAME,' MIT FOLGENDEN ATTRIBUTEN AUF');
28  WRITELN('LAUFWERK ',UNIT:2,' ANGELEGT');
29  IF DATEINAME()'INDDAT' THEN WRITELN('ANZAHL SAETZE = ',ANZS:3)
30   ELSE WRITELN('ANZAHL INDIZES = ',ANZIN:4,' TYP = ',INDEXTYP:4),
31  END;
32
33  (*********************************************************)
34  (*                  ANWEISUNGS-TEIL                      *)
35  (*********************************************************)
36
37  BEGIN
38  DATEINAME:='SDAT01'; ANZS:=10;
39
40  CREATE (UNIT,DATEINAME,ANZS,SATZ);
41  IF DFMSERROR()0
42     THEN WRITELN('FEHLER ',DFMSERROR:4,' BEI CREATE')
43     ELSE DISPLAYTEXT;
44
45  DATEINAME:='SDAT02'; ANZS:=10;
46
47  CREATE (UNIT,DATEINAME,ANZS,SATZ);
48  IF DFMSERROR()0
49     THEN WRITELN('FEHLER ',DFMSERROR:4,' BEI CREATE')
50     ELSE DISPLAYTEXT;
51
52  DATEINAME:='INDDAT'; ANZIN:=12; INDEXTYP:=0;
53  (* SORTIERT, DOPPELTE ZUGELASSEN *)
54
55  CRIND (UNIT,DATEINAME,ANZIN,INDEX,INDEXTYP);
56  IF DFMSERROR()0
57     THEN WRITELN('FEHLER ',DFMSERROR:4,' BEI CRIND')
58     ELSE DISPLAYTEXT;
59  END
```

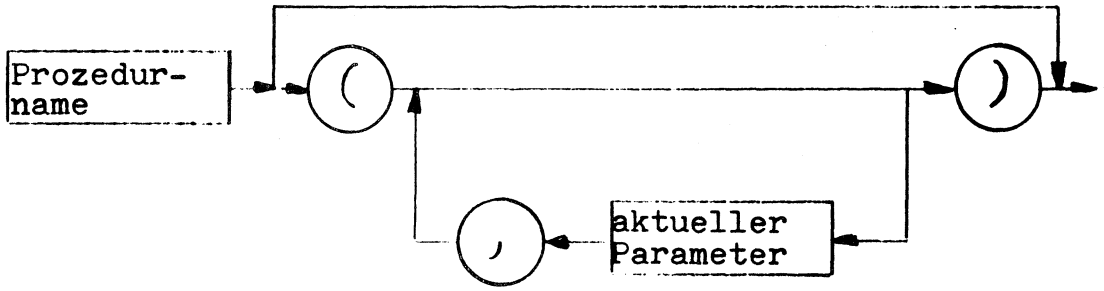
1.4. Alphabetische Liste der Aufrufe

2. Syntax der Aufrufe

2.1. Prozeduraufruf und Parameterübergabe

Die Aktivierung einer DFMS-Operation geschieht durch einen Prozeduraufruf. Die Prozeduren sind einschließlich ihrer Schnittstelle in einem System-Prolog definiert (XOSPRO), der automatisch jedem Programm, das übersetzt wird, als Vorspann dient.

Syntaxdiagramme eines Prozeduraufrufs



Beispiel:

```
CREATE (UNIT, DATEI, ANZS, SATZ)
```

Der Aufbau der aktuellen Parameterliste muß mit der formalen Parameterliste sowohl nach Anzahl als auch nach Typen überein stimmen
Es gibt zwar mehrere Möglichkeiten, Werte an eine Prozedur zu übergeben, doch hat man sich bei den DFMS-Prozeduren darauf festgelegt, daß die Parameter "call by reference" übertragen werden. Dies ist die schnellste und platzsparendste Form der Übertragung, bedeutet allerdings, daß der Prozeduraufruf keine konstanten Parameter enthalten darf. Bei der Übertragungsart "call by reference" wird die Adresse des aktuellen Parameters übertragen; die Prozedur greift über die Adresse zu den Speicherplätzen der aktuellen Parameter zu.

Wenn die Adresse des aktuellen Parameters bekannt ist, kann die Prozedur auch den Wert des Parameters ändern. Dieser Weg, der üblicherweise zur Datenrückgabe aus Prozeduren benutzt wird, schränkt somit die aktuellen Parameter auf die Verwendung von Variablen ein.

Es ist NICHT möglich:

```
CREATE (1, 'KUNDEN', 1000, SATZ)

      ! konstante Parameter sind verboten.
```

Um Variablen Anfangswerte zuzuweisen, ohne daß im Programm Kode oder Laufzeit vergeben wird, hat man bei dieser Implementierung die INITPROCEDURE eingeführt. Die Initprozeduren stehen direkt hinter den Variablenvereinbarungen und vor den Funktions- bzw. Prozedur- vereinbarungen.

Beispiel:

```
23
24  (*************************************************************)
25
26  INITPROCEDURE;
27  BEGIN
28  ENDE:=FALSE;
29  UNIT:=1;
30  SATZDATEI:='SDAT02';
31  INDEXDATEI:='INDDAT';
32  MENGEAUSWAHL:=['(',')','=','L','G','E'];
33  LT:='('; GT:=')'; EQ:='='; LE:='L'; GE:='G';
34  END;
35
36  (*************************************************************)
37
```

2.2. Funktionsaufruf

Im Prolog des erweiterten PASCAL für satzstruktuierte Dateien ist eine Erweiterung als Funktion vereinbart.

Da im Standard-PASCAL keine Möglichkeit für eine systemunterstützte Fehlerbehandlung vorgesehen ist, hat man bei dieser Implementierung die Form einer Systemfunktion gewählt, um auftretende Fehler bei DFMS-Aufrufen den Programmen zur Verfügung zu stellen.

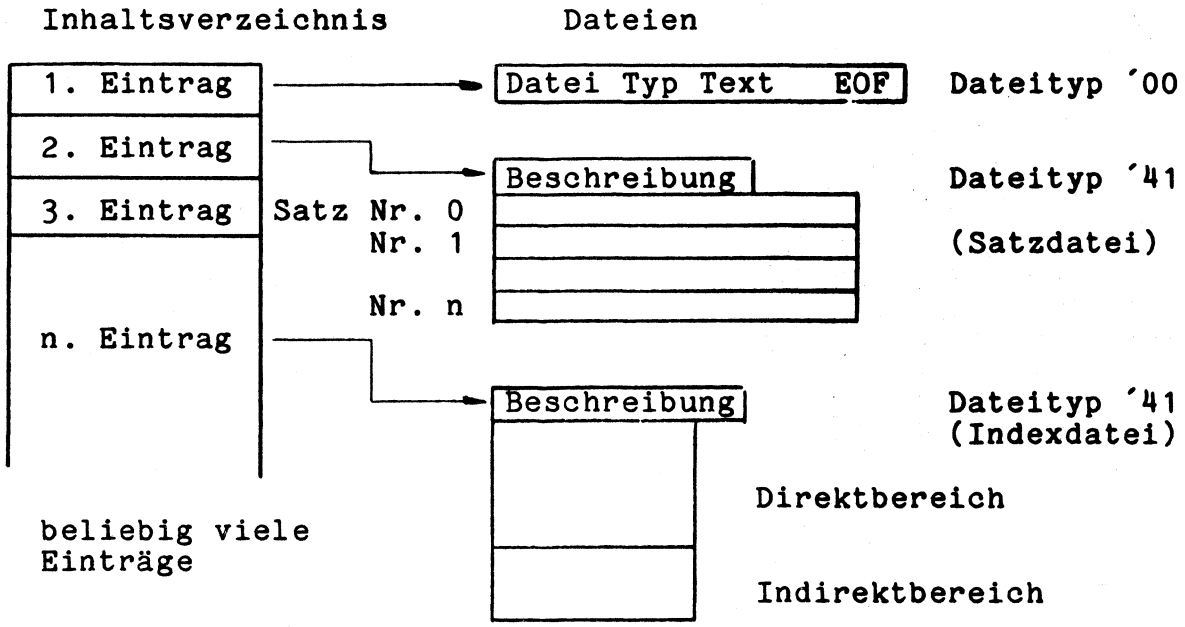
Der Ergebnistyp der Funktion ist eine Ganzzahl (INTEGER).

Aufruf in PASCAL:

```
- CASE DFMSERROR OF
            100: WRITELN ('EOF');
- IF DFMSERROR <> 0 THEN FEHLERROUTINE
            ELSE ...
- WRITELN ('DFMSERROR : ', DFMSERROR:4)
```

2.3. Semantik der Aufrufe

2.3.1. Aufbau Inhaltsverzeichnis und Dateien



Während im Inhaltsverzeichnis beliebig viele Einträge erfolgen können, wird der verwendbare Bereich einer Satzdatei bzw. einer Indexdatei beim Eintragen des Namens in das Inhaltsverzeichnis festgelegt. Jede Satz- und Indexdatei wird für sich allein eingetragen und erzeugt. Die Verkettung einer oder mehrerer Indexdateien zu einer Satzdatei erfolgt zur Laufzeit des Verarbeitungsprogramms (Verkettetes Öffnen s.a. 2.5.2).

2.4. Zugriffe zum Platteninhaltsverzeichnis

2.4.1. Autorkode

Zugriffe zum Inhaltsverzeichnis unterliegen ebenso wie Zugriffe zu Inhalten von Dateien der allgemeinen Überwachung durch das Betriebs- system.
Dabei wird sowohl der Schutz-Kode (Protection) als auch der Autorkode berücksichtigt. Beides sind Attribute zum Dateinamen. Der Schutz-Kode wird durch eine Zahl angegeben. Die Zahl setzt sich aus mehreren Summanden zusammen (s.a. 2.4.2).
Beim Autorkode unterscheidet man zwischen dem "aktuellen" Autorkode und dem Datei-Autorkode. Der aktuelle Autorkode kann gesetzt, geändert oder auch gelöscht werden.

Beispiel:

COMSYS	DIAL
*CODE,abc<cr>	*CODE,abc<cr>

!abc! steht für drei beliebige Zeichen.

Der eingetragene Kode wird durch Eingabe von CODE <cr> ohne Parameter gelöscht.

Beispiel:

COMSYS	DIAL
*CODE<cr>	*CODE<cr>

Ausgaben des Systems sind unterstrichen.

Das Setzen bzw. Löschen des Autorkodes bezieht sich nur auf einen Tabelleneintrag im Hauptspeicher. Die Tabelle ist nach Benutzern sortiert. Ein Lesezugriff auf diese Tabelle ist nicht möglich, auch nicht auf den eigenen Eintrag

2.4.1.1. Verwendung des Autorkodes

Verwendung findet der Tabelleneintrag (aktueller Autorkode) beim Anlegen einer Datei, d.h. wenn der Name der Datei in das Inhaltsverzeichnis eingetragen wird. Der Autorkode wird als letzter Zusatz in das Inhaltsverzeichnis eingetragen.

!! Eine Änderung oder ein Austragen des Datei-Autorkodes ist nicht möglich.

2.4.1.2. Beispiel:

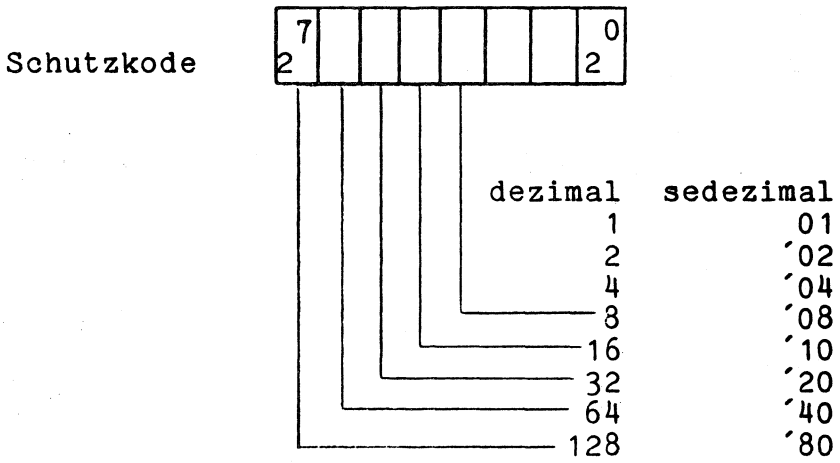
Eintrag einer Datei in das Inhaltsvereichnis

lfd.	Name	Anfang	Länge	Schutz-k.	Dateityp	Autor-k.
17	KUNDEN	17480	1236	'34	'41	DIR

An einem Benutzerplatz ist es nur möglich Einträge zu listen, bei denen der Datei-Autorkode mit dem aktuellen Autorkode (Tabelleneintrag) übereinstimmt. Dagegen werden am Masterplatz alle Dateien gelistet. Bei Zugriffen zum Inhaltsverzeichniseintrag einer Datei wird der Fehler Nr. 65 (Datei nicht vorhanden), gemeldet, wenn der aktuelle Autorkode nicht mit dem Datei-Autorkode übereinstimmt. Lese- bzw. Schreibzugriffe zu Dateien werden unter Berücksichtigung des Schutz-Kodes (Protection) ausgeführt (siehe auch 2.4.2).

2.4.2. Schutz-Kode setzen bzw. ändern

Der Schutz-Kode besteht aus einer Zahl, die sich aus 2er Potenzen als Summanden zusammensetzt. Die Zahl wird je nach Verwendung als dezimale Zahl oder sedezimale Zahl dargestellt. Die interne Speicherung im Inhaltsverzeichnis einer Magnetplatte erfolgt in einen Byte. Daraus ergeben sich acht Möglichkeitn, bzw. Summen der acht Möglichkeiten.



Die Anzeige des Schutz-Kode beim Listen des Inhaltsverzeichnisses erfolge sedezimal, ebenso wie die Eingabe des Parameters beim Dialogkommando.

2.4.2.1. Beispiel

```
SERV,LIST,M,0,0,SERVIC<cr>
```

lfd.	Name	Anfang	Länge	Schutz-k.	Dateityp	Autor-k.
1	SERVIC	48	512	´05		´32

Die Kennzeichnung, daß es sich um eine sedezimale Darstellung handelt, erfolgt durch das Hochkomma vor der Zahl.

Beispiel:

```
PROT,1,TEXT,´10<cr>
```

Auch hier erfolgt die Kennzeichnung durch das Hochkomma.
Allerdings sei hier darauf hingewiesen, daß die Parameterdekodierung des Kommandointerpreters beim Kommando PROT immer eine sedezimale Zahl als dritten Parameter unterstellt.
Somit kann das Hochkomma auch entfallen.

Da PASCAL keine sedezimalen Zahlen kennt, erfolgt die Angabe von (p) als Dezimalzahl.

Aufruf in PASCAL:

```
PROT (u,f,p)
```

u = unit/Laufwerk
f = filename/Dateiname
p = protectioncode/Schutzkode

Die Typen der aktuellen Parameter (u) und (p) müssen INTEGER sein.
! Auch Subrange von Integer ist nicht erlaubt. Der Typ des Parameters f muß ARRAY [1..6] OF CHAR sein. Die aktuellen Parameter müssen Variable sein, da die Übertragung zur Prozedur "call by reference" erfolgt.

Die Summanden haben folgende Bedeutung:

1	Austragen aus dem Inhaltsverzeichnis nicht erlaubt (siehe auch 2.4.3)
2	zur Zeit frei
4	Datendatei (auch Quellkode-Dateien sind als Datendateien zu betrachten)
8	zur Zeit frei
16	Leseprivat
32	Schreibprivat
64	zur Zeit frei
128	zur Zeit frei

Die beiden Summanden 16 und 32 (lese- bzw. schreibprivat) sind nur wirksam, wenn auch der Datei-Autorkode gesetzt ist.
Dateien ohne Autorkode sind grundsätzlich öffentlich. Sind sowohl der Datei- Autorkode als auch die Summanden 16 und 32 gesetzt, so erfolgt die Fehlermeldung Nr. 65 (Datei nicht vorhanden) schon beim Öffnen der Datei (siehe auch 2.5), wenn der "aktuelle Autorkode" nicht mit dem Datei-Autorkode übereinstimmt.
Ist einer der Summanden 16 oder 32 nicht gesetzt, so ist das Öffnen grundsätzlich möglich. Erst beim Lese- bzw. Schreibzugriff erfolgt eine Überprüfung mit eventueller Fehlermeldung Nr. 68 (verlangte Zugriffsart nicht erlaubt).

Mögliche Fehler: 1,3,4,65,68

2.4.3. Dateiname ändern

Es sind grundsätzlich alle Zeichen als Dateiname erlaubt. Empfehlenswert ist trotzdem, den Dateinamen auf druckbare Zeichen zu beschränken [.0..9,´A´..´Z´,und Sonderzeichen.].

Aufruf in PASCAL:

```
ALTER (u,fa,fn)
```

u = unit/Laufwerk
fa = filename/Dateiname alt
fn = filename/Dateiname neu

Die Typen der beiden Parameter fa und fn müssen ARRAY [.1..6.] OF CHAR sein. Der Typ von (u) muß INTEGER sein. Die aktuellen Parameter müssen Variable sein.

Mögliche Fehler: 1,3,4,65,68

2.4.4. Dateiname austragen

Eine Datei gilt als nicht mehr vorhanden, wenn der Dateiname aus dem Inhaltsverzeichnis ausgetragen worden ist (gelöscht). Der Bereich der Platte, der die Datei hält, wird nicht verändert. Ebenso bleiben die Zusatzangaben im Inhaltsverzeichnis (phys. Anfang und Länge) erhalten.
Die Datei kann aber nicht mehr gelistet werden. Es kann auch nicht mehr zu den Daten zugegriffen werden.

Bei Eintragen einer neuen Datei wird ein freier Platz im Inhaltsverzeichnis bestimmt, hinter dem sich ein möglichst exakt passender Freibereich der Platte verbirgt.
Erst wenn kein freier Platz im Inhaltsverzeichnis gefunden wurde, wird die Datei am Ende des Inhaltsverzeichnisses eingetragen und damit Platz hinter der physisch zuletzt angelegten Datei reserviert.

Somit wird eine permanente Reorganisation des Inhaltsverzeichnisses und damit auch des Plattenspeichers vorgenommen.

Aufruf in PASCAL:

```
KILL (u,f)
```

u = unit/Laufwerk
f = filename/Dateiname

Der Typ des Parameters (u) muß INTEGER sein, der Typ des Parameters (f) ARRAY [1..6] OF CHAR. Die aktuellen Parameter müssen Variable sein.

Mögliche Fehler: 1, 3, 4, 65, 68

2.4.5. Satzdatei eintragen und vorbesetzen

Ein Eintrag im Inhaltverzeichnis kann implizit in einem Aufruf (Kommando) enthalten sein.

z. B. OBJ,u,f

Objektcode erzeugen und in die Datei (f) auf dem Laufwerk (u) übertragen. Vorher wird der Name (f) in das Inhaltsverzeichnis der Platte (u) eingetragen.

Oder aber der Eintrag wird unter Angabe der Struktur explizit durchgeführt. Je nach Aufruf wird ein bestimmter Dateityp vom System gesetzt.

Aufruf in PASCAL:

```
CREATE (u,f,n,rec)
```

u	=	unit/Laufwerk
f	=	file/Dateiname
n	=	Anzahl Sätze
rec	=	record/Satz

Während die Typen der ersten drei Parameter festliegen (u = INTEGER, f = ARRAY [1..6] OF CHAR, n = INTEGER) bestimmt der 4. Parameter durch seinen Typ und durch seine Struktur den Aufbau der Sätze (siehe Beispiel PDFS01).
Die maximale Anzahl der Sätze ist auf 32767 (1 <= n <= 32767) begrenzt.
Der maximal zur Verfügung stehende Platz pro Satz beträgt 32765 Byte (1 <= Länge(rec) <= 32765). Eine Kalkulation der benötigten Länge ist nur bedingt möglich und auch nicht sinnvoll.

2.4.5.1. Beispiel: Eintrag einer Satzdatei

Nach Ausführung der Prozedur ergibt sich folgender Eintrag im Inhaltsverzeichnis der Platte:

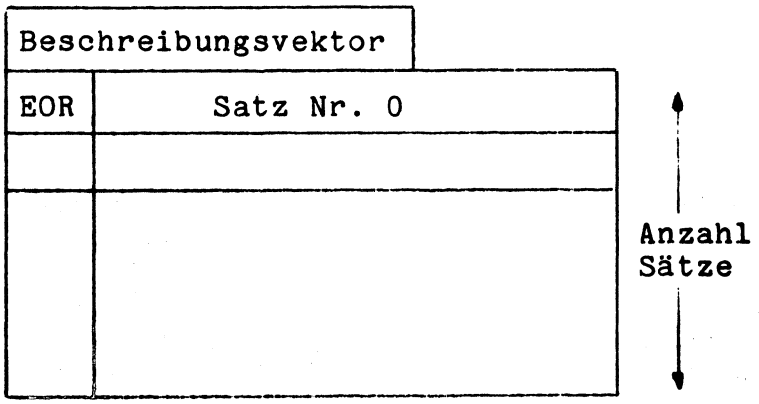
CREATE (u, f, n, rec)

lfd.Nr.	Name	Anfang	Länge	Schutz-k.	Dateityp	Autork.
nr.	f	anf.	l.	´34	´41	abc

nr. wird vom System vergeben
f aktueller Inhalt des 2. Parameters zum Zeitpunkt des Aufrufes
anf wird vom System vergeben
l wird vom System errechnet und vergeben (abhängig von n und rec)
´34 schreib- und leseprivat und Datendatei
´41 DFMS - Datei
abc wurde dem aktuellen Autorkode entnommen (s.a. 2.4.1)

Neben dem Eintrag im Inhaltsverzeichnis der Platte erfolgt eine Vorbesetzung der Datei.

2.4.5.2. Beispiel: Aufbau einer Satzdatei



Der Beschreibungsvektor enthält neben internen Merkern für Release Nr und Typangabe (Satz/Indexdatei) noch den Freizeiger (FZ) für die Freisatzverwaltung bei Eintragung von Indizes in eine zugeordnete Indexdatei (siehe auch 2.5.2 und 2.8.4).

Alle EOR Zeiger (Füllhöhenangabe) werden auf 0 gesetzt.

Mögliche Fehler: 1, 3, 4, 65, 69, 104

2.4.6. Indexdatei eintragen und vorbesetzen

Der Eintrag einer Indexdatei in das Inhaltsverzeichnis ist von dem Eintrag einer Satzdatei nicht anhand des Dateityps zu unterscheiden. Beide haben als Dateityp '41. Es ist somit empfehlenswert, ein Zeichen des Dateinamens als Kennzeichen zu verwenden.

z. B. KUNDS1 als 1. Kundensatzdatei
KUNDI1 als 1. Kundenindexdatei

Aufruf in PASCAL:

```
CRIND (u,f,n,k,t)
```

u = unit/Laufwerk
f = filename/Dateiname
n = Anzahl der Indizes
i = Index
t = Typ

Bei diesem Aufruf liegen die Typen der ersten drei und der Typ des letzten Parameters fest. Aus dem Typ des vierten Parameters (k) ergibt sich der Platzbedarf des Indizes.

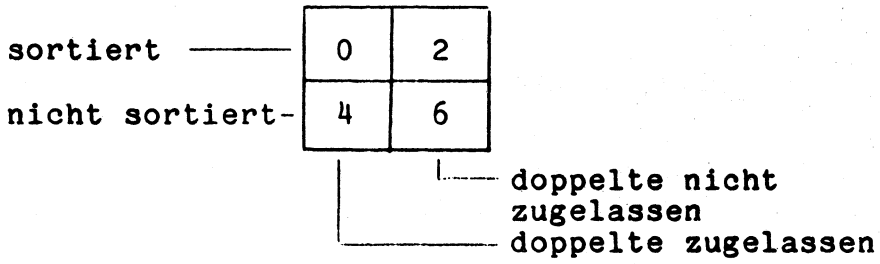
Die Parameter (u), (n) und (t) sind vom Typ INTEGER; der Parameter (f) ist vom Typ ARRAY [1..6] OF CHAR. Der Typ einer Indexdatei legt das Verhalten des Moduls DFMS bei Eintragungen, Austragungen und Änderungen fest. Die Zeit, die der Modul DFMS benötigt, um eine Eintragung, Austragung oder Änderung durchzuführen bzw. um einen Index bei einem Suchauftrag zu finden, ist vom Typ abhängig.

2.4.6.1. Zugriffsmechanismen

Die Anzahl der Indizes ist auf 32767 begrenzt. Der Platzbedarf des Indizes - bestimmt durch den Parameter k - darf nicht mehr als 119 Byte betragen. Der Typ wird durch eine Dezimalzahl angegeben, die sich aus folgender Formel ergibt:

t = z * 16

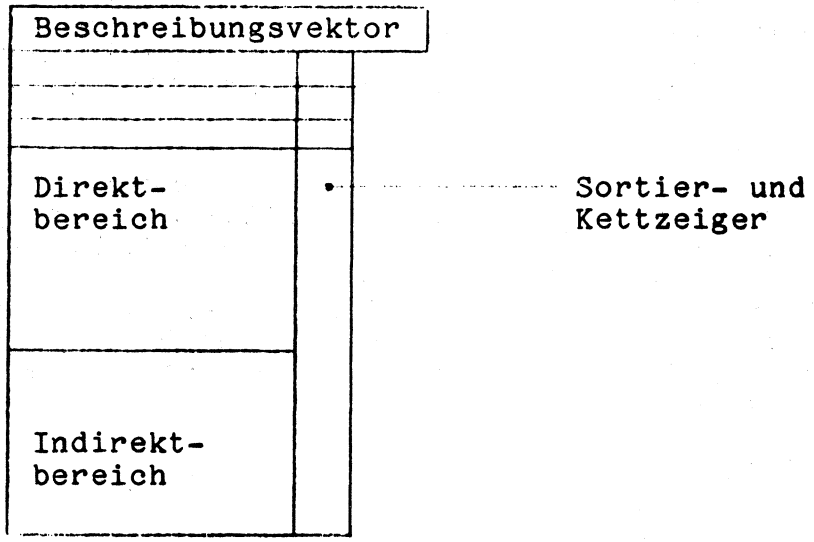
Der Wert für z kann folgender Matrix entnommen werden:



! Vorsicht, das Eintragen von Indizes in eine Datei vom Typ sortiert bedeutet für den Modul DFMS mehr Zeitaufwand, da neben dem Hash-Zeiger noch die Vorwärts-/ Rückwärtsverkettung vorgenommen werden muß. Häufig ist es besser, eine Datei vom Typ unsortiert zu verwenden und immer dann zu sortieren (KEYSORT, s. a. 2.8.8.1), wenn man wirklich eine Sortierung benötigt (z. B. Listen drucken).

Der Modul DFMS erzeugt die Fehlermeldung Nr. 103 (Doppelter Index), wenn ein Index eingetragen werden soll und ein identischer Index in der Datei mit Kennzeichnung 32 oder 96 existiert.

2.4.6.2. Beispiel: Aufbau einer Indexdatei



Die Größe des Direktbereiches (d) ergibt sich aus der max. Anzahl der Indizes. Und zwar wird vom System die nächst kleinere 2er Potenz gewählt (2er Potenzen 1,2,4,8,16 ... 1024,2048,4096...). Die Größe des Indirektbereiches ergibt sich aus der Differenz zwischen n (Anzahl der Indizes) und Größe des Direktbereiches (d).

Die Bedeutung des Hash-Zeigers kann der Benutzeranleitung XOS-Modul DFMS entnommen werden.

Mögliche Fehler: 1, 3, 4, 65, 69, 104

2.5. Dateien öffnen und schließen

Da bei Zugriffen zu Dateien immer zu den tatsächlich reservierten physischen Bereichen der Platte zugegriffen werden muß - die Angaben hierüber aber im Inhaltsverzeichnis der Platte stehen - überträgt das System beim "Öffnen" alle relevanten Informationen aus dem Inhaltsverzeichnis in den Arbeitsspeicher (Opentabelle) .

Dieser Ladevorgang - sprich, das Öffnen von Dateien - ist nur möglich, wenn der aktuelle Autorkode gleich dem Datei-Autorkode ist, oder wenn die Datei lese- oder/und schreiböffentlich ist.
Ein Zugriff auf die Opentabelle ist von PASCAL aus nicht möglich. Das Schließen einer Datei heißt, daß der beim Öffnen belegte Bereich in der Opentabelle wieder freigegeben wird.

2.5.1. Öffnen einer Satz- oder Indexdatei einzeln

Beim Öffnen vergibt der Modul DFMS eine Ganzzahl, die man als Arbeitsnummer bezeichnet. Ein Zugriff zu einer Datei geschieht immer durch die Angabe der Arbeitsnummer.
Die Arbeitsnummer wird in einer Variablen vom Typ INTEGER geführt. Eine Veränderung der Variablen nach dem Öffnen und vor dem Zugriff führt zu einem Fehlverhalten des Systems und ist deshalb verboten.
Der Satz- bzw. Indexzeiger steht nach dem Öffnen auf 0 bzw. auf dem Index mit dem niedrigsten Wert.

Aufruf in PASCAL:

```
OPENDIREKT (u, f, w)
```

u = unit/Laufwerk
f = filename/Dateiname
w = worknr./Arbeitsnummer

Die Typen der Parameter (u) und (w) sind beide INTEGER, der Typ des Parameters (f) ist ARRAY [1..6] OF CHAR.

2.5.1.1. Beispiel: Öffnen einer Satz- oder Indexdatei

```
20
21   (*************************************************************)
22
23   INITPROCEDURE;
24   BEGIN
25   ENDE:=FALSE;
26   UNIT:=1;
27   SATZDATEI1:='SDAT01';
28   SATZDATEI2:='SDAT02';
29   INDEXDATEI:='INDDAT';
30   END;
31
32   (*************************************************)
33

92
93   (*************************************************)
94   (*              ANWEISUNGSTEIL                   *)
95   (*************************************************)
96
97   BEGIN
98
99   OPENDIRECT(UNIT,SATZDATEI1,ARB1);
100  IF DFMSERROR()0 THEN FEHLER('OPENDIRECT
```

Mögliche Fehler: 1,3,65,72,104,105

2.5.2. Öffnen einer Satzdatei verkettet mit einer Indexdatei

Um einen indexabsoluten oder einen indexsequentiellen Zugriff durchführen zu können, müssen sowohl die Satzdatei als auch die Indexdatei geöffnet werden. Die beiden Dateien werden über eine gemeinsame Arbeitsnummer verknüpft. Der Indexzeiger steht nach dem Öffnen auf dem Index mit dem niedrigsten Wert.

Aufruf in PASCAL:

```
OPENINDEXED (us, fs, ui, fi, w)
```

us = unit/Laufwerk der Satzdatei
fs = file/Dateiname der Satzdatei
ui = unit/Laufwerk der Indexdatei
fi = file/Dateiname der Indexdatei
w = worknumber/Arbeitsnummer

Die Parameter us und ui sind vom Typ Integer, die Parameter (fs) und (fi) sind vom Typ ARRAY [1..6] OF CHAR.

2.5.2.1. Beispiel: Verkettetes Öffnen

```
20
21   (*************************************************************)
22
23   INITPROCEDURE;
24   BEGIN
25   ENDE :=FALSE;
26   UNIT :=1;
27   SATZDATEI1 :='SDAT01';
28   SATZDATEI2 :='SDAT02';
29   INDEXDATEI :='INDDAT';
30   END;
31
32   (*************************************************************)
33

92
93   (*************************************************************)
94   (*                  ANWEISUNGSTEIL                           *)
95   (*************************************************************)
96
97   BEGIN
98
99   OPENDIRECT(UNIT,SATZDATEI1,ARB1);
100  IF DFMSERROR()0 THEN FEHLER('OPENDIRECT     ') ELSE
101  BEGIN
102  OPENINDEXED(UNIT,SATZDATEI2,UNIT,INDEXDATEI,ARB2);
103  IF DFMSERROR()0 THEN FEHLER('OPENINDEXED    ') ELSE
104  BEGIN
105
106  REPEAT
107  MENUEAUSGABE;
```

Mögliche Fehler: 1,3,65,72,104,105

2.5.3. Schließen der Dateien

Um einem Überlauf der Opentabelle vorzubeugen, kann man Dateien schließen, zu denen im weiteren Programmverlauf nicht mehr zugegriffen werden soll. Dateien während der Programmausführung mehrfach zu öffnen und zu schliessen, um eine geringe Belegung der Opentabelle zu erreichen, sollte man vermeiden, da das öffnen von Dateien bei hoher Belegung des Inhaltsverzeichnisses zeitaufwendig ist.

Aufruf in PASCAL:

```
CLOSE (w)
```

oder

```
CLOSEALL
```

w = worknumber /Arbeitsnumer

Bei Programmende (END) oder durch Fehler im Programm, die zur Rückkehr in den Dialog führen, werden alle Dateien des betreffenden Benutzers geschlossen.

2.6. Zugriffe zu Satzdateien

2.6.1. Lesen eines Satzes

Der Lesezugriff erfolgt immer auf den sogenannten "aktuellen Satz". Der aktuelle Satz ist durch den Wert des Satzzeiger in der Opentabelle definiert. Nach Abschluß des Lesens kann das System den Satzzeiger direkt durch Addition, bzw. indirekt über logisch-sequentielles Fortschalten erhöhen (Zugriff auf die dazugehörende Indexdatei).

Aufruf in PASCAL:

```
READS (w, rec) ohne Fortschaltung

READNEXT (w, rec) mit Fortschaltung
```

w = worknumber/Arbeitsnummer
rec = record/Satz

Der Zugriff zu einem leeren Satz führt zu dem Fehler 101. Ebenso wird Fehler 101 angezeigt, wenn die Variable (rec) mehr Bytes aufnehmen kann, als im Satz beschrieben sind.

Mögliche Fehler: 1,3,5,68,100,101,102

2.6.2. Schreiben eines Satzes

Ebenso wie der Lesezugriff, erfolgt der Schreibzugriff immer auf den aktuellen Satz. Der aktuelle Satz wird durch den Satzzeiger der Opentabelle bestimmt. Dieser ist explizit änderbar (s. a. 2.7.2 SELDIRECT) oder implizit durch ENTERKEY setzbar. Der Aufruf ENTERKEY besorgt die Nummer des Freisatzes aus der beim verketteten Öffnen mitgeöffneten Satzdatei und besetzt damit den Satzzeiger. Da ENTERKEY somit bei jedem Eintrag den Satzzeiger neu setzt, ist es unsinnig einen sequentiellen Schreibzugriff auf eine "verkettete Datei" anzuwenden.

Aufruf in PASCAL:

```
WRITES (w, rec) ohne Fortschaltung

WRITENEXT (w, rec) mit Fortschaltung
```

w = worknumber/Arbeitsnummer
rec = record/Satz

Ist der zu beschreibende Satz leer, wird ab Anfang des Satzes geschrieben, sonst direkt hinter der letzten beschriebenen Position.

Mögliche Fehler: 1,3,4,68,100,101,102

2.6.3. Lesen und Sperren eines Satzes

Immer dann, wenn zwei Benutzer (Programme, Jobs etc.) gleichzeitig einen Satz zum Zwecke des abhängigen Änderns lesen wollen, muß der Satz nach dem Lesen gesperrt werden.

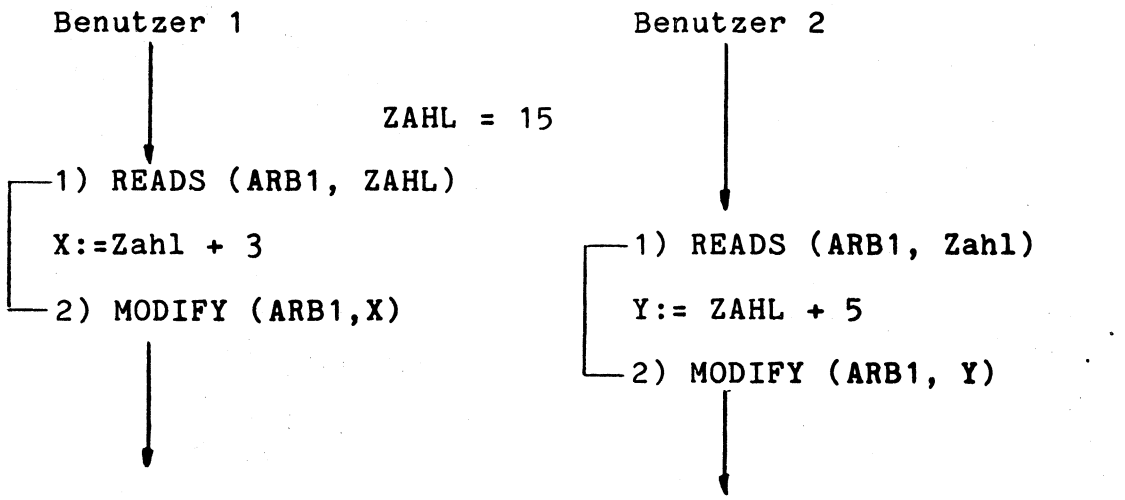
Aufruf in PASCAL:

```
UPDATE (w, rec)
```

w = worknumber/Arbeitsnummer
rec = record/Satz

Der durch UPDATE gelesene Satz bleibt solange für andere Benutzer gesperrt, bis der sperrende Benutzer, durch Fortschaltung des Satzzeigers, den Satz wieder freigibt. Der Wert des Satzoffsetzeigers wird vor Ausführung des UPDATE Aufrufs gespeichert. Dieser Wert wird beim MODIFY-Aufruf als Anfangsposition benötigt. Eine ausführliche Beschreibung der Zeigerbehandlung (z. B. des Satzoffsetzeigers) kann der Benutzeranleitung XOS-Modul DFMS entnommen werden.

2.6.3.1. Beispiel einer fehlerhaften Datenänderung



Der Wert des Feldes Zahl sollte nach Beendigung der beiden Zugriffe 23 sein. Dieses ist nur durch eine Zugriffssperre zwischen den Zeitpunkten 1) und 2) zu erreichen.

Mögliche Fehler : 1,3,4,68,100,101,102

2.6.4. Modifizieren eines Satzes

Die Funktion MODIFY entspricht im wesentlichen der Funktion WRITE. Besonderheit der Funktion ist die Möglichkeit Sätze oder Satzteile zu überschreiben. Während der WRITE Aufruf immer ab dem ersten freien Byte schreibt, beginnt die Übertragung durch den MODIFY Aufruf ab der Position im Satz, die durch den UPDATE-Aufruf gespeichert wurde.

Aufruf in PASCAL:

```
MODIFY (w, rec) ohne Fortschaltung

MODNEXT (w, rec) mit Fortschaltung
```

w = worknumber/Arbeitsnummer
rec = record/Satz

Durch die Fortschaltung mit MODNEXT wird die Satzsperre aufgehoben.

Mögliche Fehler: 1,3,4,68,100,101,102

2.6.5. Löschen eines Satzes

Aufruf in PASCAL:

```
DELETE (w)
```

w = worknumber/Arbeitsnummer

Der aktuelle Satz wird gelöscht, d. h. der pro Satz vorhandene Füllhöhenzeiger (EOR) bekommt den Wert 0. Ein Zugriff auf die Daten ist nicht mehr möglich. Eine Verwendung gelöschter Sätze ist nach Ablauf der Prozedur FILEREORG oder durch Direktzugriff über die Satznummer möglich. Der Aufruf schaltet weder Satz- noch Indexzeiger weiter.

Mögliche Fehler: 1,3,4,68,100,102

2.7. Satzanwahl

2.7.1. Sequentielle Weiterschaltung

Die sequentielle Weiterschaltung kann sowohl implizit in Lese- oder Schreibaufrufen erfolgen, als auch durch explizites Aufrufen der Funktion NEXT.

Aufruf in PASCAL:

```
NEXT (w)
```

w = worknumber/Arbeitsnummer

Die Arbeitsnummer darf sich auf eine Satzdatei, auf eine Indexdatei oder auch auf eine verkettete Datei beziehen.
Im ersten Fall wird auf den physisch nächsten Satz geschaltet.
Im zweiten Fall wird erst der logisch nächste Index angewählt und im dritten Fall wird erst der logisch nächste Index angeählt und dann der Satzzeiger auf den Wert der Satznummer gesetzt, die als Referenzangabe hinter dem Index steht.

Dieser explizite Aufruf ist wichtig, weil die implizit im READNEXT, WRITENEXT und MODNEXT enthaltenen Fortschaltung nur wirksam wird wenn der Lese- bzw. Schreibaufruf erfolgreich war. Im Falle eines Fehlers wird nicht weitergeschaltet.

Mögliche Fehler: 100, 105

2.7.2. Direkte Anwahl über eine Satznummer

Die direkte Anwahl eines Satzes geschieht durch Setzen des Satzzeigers auf den Wert der im Aufruf angegebenen Satznummer.

Aufruf in PASCAL:

```
SELDIRECT (w, snr)
```

w = worknumber/Arbeitsnummer
snr = Satznummer

Beide Parameter müssen vom Typ Integer sein. Die Datei, die sich hinter der Arbeitsnummer (w) verbirgt, muß eine Satzdatei sein und mit OPENDIRECT geöffnet sein.

Mögliche Fehler: 100, 102, 104

2.7.2.1. Indirekte Anwahl über einen Index

Der angegebene Index muß an allen Stellen siqnifikant sein. Es ist keine Maskenfunktion vorgesehen (s.a. 2.8.2 Suchen eines Indizes). Wurde der Index gefunden, wird die Referenzangabe (Satznummer) an den Satzzeiger weitergegeben.

Aufruf in PASCAL:

```
SELINDEXED (w, i)
```

w = Arbeitsnummer
i = Index

Der Typ des Parameters (w) ist Integer. Der Typ des 2. Parameters muß der Angabe bei CRIND (s.a. 2.4.5 Indexdatei eintragen) entsprechen. Die Arbeitsnummer (w) muß sich auf eine verkettete Datei beziehen (OPENINDEXED s.a. 2.5.2)

Mögliche Fehler: 102, 103

2.8. Zugriffe zu Indexdateien

Bei den folgenden Aufrufen ist zu beachten, daß einige Aufrufe sich sowohl auf eine einzeln geöffnete Indexdatei als auch auf verkettete Dateien beziehen können.

2.8.1. Lesen eines Indizes

Es wird der aktuelle Index mit der dazugehörenden Satznummer gelesen und den Parametern (i) und (snr) zugewiesen.

Aufruf in PASCAL:

```
GETKEY (w, i, snr) ohne Fortschaltung

GETKNEXT (w, i, snr) mit Fortschaltung
```

w = worknumber/Arbeitsnummer
i = Index
snr = Satznummer

Die Parameter (w) und (snr) sind vom Typ Integer. Der Typ des Parameters (i) muß identisch mit dem Typ des Parameters (i) bei dem Aufruf CRIND sein, der die Indexdatei erzeugt hat.

Mögliche Fehler : 1,3,5,100,102

2.8.2. Suchen eines Indexes

Diese Form der impliziten Satzanwahl wird immer dann angewandt, wenn der zu suchende Index nicht vollständig bekannt ist.
Neben der Möglichkeit, auf einen Index zu positionieren, der kleiner, größer, kleiner oder gleich bzw. größer oder gleich als der angegebene Index ist, können Sternchen "*" als Sonderzeichen verwendet werden.
Die Stellen, die im Vergleichsindex mit Sternchen markiert sind, werden beim Suchen nicht berücksichtigt (s.a. Beispiel PDFS04). Die Sternchen haben nur dann eine Sonderbedeutung, wenn bei der Generierung (SERV,MODULE,MDFMS) der letzte Parameter entsprechend gewählt wird (s.a. 1.2.1 Generierung und Benutzeranleitung Betriebssystem Modul DFMS Kap. 1.2.1)

Aufruf in PASCAL:

```
SEKEY (w, i1, b, i2)
```

w = worknumber/Arbeitsnummer
i1 = Vergleichsindex
b = Vergleichsoperator
i2 = Hilfsvariable

Der 1. Parameter (w) ist vom Typ INTEGER, der 3. Parameter (Vergleichsoperator b) ist vom Typ CHAR. Die beiden Parameter i1 und i2 müssen identisch mit dem Parameter sein, der bei der Erzeugung der angesprochenen Indexdatei verwendet wurde.

2.8.2.1. Vergleichsoperatoren

Bei Aufruf der Funktion SEKEY wird der Index nach i2 übergeben, der die vorgegebene Vergleichsbedingung am besten erfüllt.
Der Satzzeiger wird außerdem auf den Wert des zu i2 gehörenden Satzes gesetzt, so daß direkt danach ein Zugriff zu dem Satz erfolgen kann.

Für b ist zulässig:

```
'<': i1 < i2
'>': i1 > i2
'=': i1 = i2
'L': i1 <= i2
'G': i1 >= i2
```

Mögliche Fehler: 1, 3, 5, 102, 103

2.8.3. Anwahl des ersten Indizes

Diese Funktion ersetzt einen Aufruf von SEKEY mit einem Index, der kleiner ist, als alle in der Datei enthaltenen.
Die Satznummer des angewählten Indizes wird in den Satzzeiger übertragen

Aufruf in PASCAL:

```
FIRST (w)
```

w = worknumber/Arbeitsnummer

Mögliche Fehler: 102

2.8.4. Eintragen eines Indizes

Die Funktion ENTERKEY kann nur auf verkettete Dateien angewendet werden (OPENINDEXED). Nach Lesen des Freizeigers aus der Satzdatei wird, dieser um 1 erhöht in die Satzdatei, zurückgeschrieben. Der gelesene Wert wird zusammen mit dem Index in die Indexdatei eingetragen und auch dem Satzzeiger zugewiesen. Somit kann ein danach folgender WRITES Aufruf den Satz direkt beschreiben (s.a. Beispiel PDFS02)

Aufruf in PASCAL:

```
ENTERKEY (w, i)
```

w = worknumber/Arbeitsnummer
i = Index

Ob neben dem HASH-Zeiger auch noch die Sortierzeiger gesetzt werden müssen, wird anhand des Typs der Indexdatei entschieden.

Beim Typ Doppelte nicht zugelassen (32 und 96) wird der Fehler 103 gemeldet, wenn ein identischer Index beim Eintragen gefunden wird

2.8.4.1. Eintrag mit Vorgabe der Satznummer

Aufruf in PASCAL:

```
ENKEYANDNUMBER (w, i, snr)
```

w = worknumber/Arbeitsnummer
i = Index
snr = Satznummer

Diese Möglichkeit erlaubt die explizite Vorgabe der Satznummer. Die zu der Arbeitsnummer (w) gehörende Indexdatei muß mit "OPENDIRECT" geöffnet worden sein.

2.8.4.2. Sortiertes Eintragen

Unabhängig vom Typ der Indexdatei kann die Sortierverkettung durch die beiden nächsten Aufrufe erfolgen.

Aufruf in PASCAL

```
SORKEY (w, i)

SORKNUM (w, i, snr)
```

w = worknumber/Arbeitsnummer
i = Index
snr = Satznummer

Die beiden Aufrufe entsprechen im weiteren Ablauf vollständig den Aufrufen ENTERKEY und ENKEYANDNUMBER.

Mögliche Fehler : 1,3,4,5,68,100,102,109

2.8.5. Austragen eines Indizes

Der Aufruf UNKEY erlaubt zwei unterschiedliche Vorgehensweisen. Abhängig vom ersten Zeichen des Indizes wird entweder der aktuelle Index gelöscht oder aber erst der angegebene Index gesucht und dann gelöscht.

Aufruf in PASCAL:

```
UNKEY (w, i)
```

w = Arbeitsnummer
i = Index

- 1. Zeichen des Indizes gleich CHR(0)
  löschen des aktuellen Indizes

- 1. Zeichen des Indizes ungleich CHR(0)

  löschen des aktuellen Indizes

Bei doppelten Indizes, wie z. B. in einer Indexdatei über ´Namen´ oder ´Orte´, sehr wohl vorkommen können, ist das gezielte Löschen des 2. oder jedes weiteren Indizes nur durch UNKEY (aktueller Index) möglich.
Die Funktion SEKEY positioniert immer auf den Index, der in der zeitlichen Reihenfolge als erster eingetragen wurde.

Die beim Austragen freiwerdenden Plätze können erst nach Ausführung der Prozedur KEYREORG wieder verwendet werden.

Mögliche Fehler: 102, 103

2.8.6. Ändern eines Indizes

Die Funktion RENAMEKEY kann durch eine Anzahl von schon bisher behandelten Prozeduren nachgebildet werden. Zu beachten ist - wie bei der Funktion UNKEY - die Möglichkeit des aktuellen Indizes.

Aufruf in PASCAL:

```
RENAMEKEY (w, ia, in)
```

w	=	Arbeitsnummer
ia	=	alter Index
in	=	neuer Index

Die Prozedur setzt setzt sich aus folgenden Aufrufen zusammen:

- SEKEY
- GETKEY
- UNKEY
- ENKEYANDNUMBER

Die Fehlerbehandlung erfolgt in den genannten Schritten der Teilfunktionen, so daß z.B. auch Dateiüberlauf beim Umbenennen eines Indizes auftreten kann.

Mögliche Fehler: 100, 102, 103

2.8.7. Verbinden zweier Indizes

Durch die Funktion CONNECTKEY wird einem Satz ein zweiter Index zugeordnet. Der zweite Index wird in eine Datei eingetragen, die direkt geöffnet wurde (OPENDIRECT ui, fi, w1).

Aufruf in PASCAL:

```
CONNECTKEY (w1, i1, w2, i2)
```

w1	=	Arbeitsnummer der direkt geöffneten Indexdatei
i1	=	neuer Index
w2	=	Arbeitsnummer der direkt oder verkettet geöffneten Indexdatei
i2	=	vorhandener Index

Der Index (i1) wird in die unter der Arbeitsnummer (w1) geöffneten Indexdatei eingetragen. Gleichzeitig wird auch die Satznummer eingetragen, die vorher aus der Indexdatei (w2) besorgt wurde.
Die Verwendung des aktuellen Indizes bei (w2, i2) erfolgt wie bei UNKEY und RENAMEKEY.
Die Funktion CONNECTKEY setzt sich aus folgenden Teilfunktionen zusammen:

- SEKEY
- GETKEY
- ENKEYANDNUMBER

Wie auch bei RENAMEKEY erfolgt die Fehlerbehandlung in den Schritten der einzelnen Funktionen.
Auch hier ist zu beachten, daß bei doppelten Indizes immer der in der zeitlichen Reihenfolge als erster eingetragene durch SEKEY gefunden wird.

Mögliche Fehler: 100, 102, 103

2.8.8. Hilfsroutinen

Die Hilfsroutinen benötigen zwei Sortierdateien (SFILE0 u. SFILE1) auf der Systemresidenz. Die Größe der beiden Dateien muß gleich sein und berechnet sich nach folgender Angabe:

Je Index einer zu verarbeitenden Indexdatei werden die Indexlänge + 2 gerundet auf die nächste 2er Potenz benötigt. Außerdem müssen die Dateien mindestens die physikalische Länge der zu bearbeitenden Indexdateien haben.

2.8.8.1. Sortieren einer Indexdatei

Aufruf in PASCAL:

```
KEYSORT (u, f)
```

u = unit/Laufwerk
f = filename /Dateiname

Der Typ des Parameters u ist Integer, der des Parameters f ist ARRAY [1..6] OF CHAR.

Nach Ablauf dieser Funktion sind alle Indizes der Datei (f) sequentiell zugreifbar.
Es werden die Dateien SFILE0 und SFILE1 auf der Systemplatte benötigt.

Mögliche Fehler: 1, 3, 4, 65, 68, 72

2.8.8.2. Reorganisieren einer Indexdatei

Aufruf in PASCAL:

```
KEYREORG (u1, f1, u2, f2)
```

u1 = unit/Laufwerk Quelle
f1 = file/Dateiname Quelle
u2 = unit/Laufwerk Ziel
f2 = file/Dateiname Ziel

Es ist erlaubt, daß Quelle und Ziel identisch sind. Die angegebene Indexdatei wird dahingehend reorganisiert, daß die Plätze innerhalb der Indexdatei, die durch UNKEY oder RENAMEKEY frei geworden sind, wieder verwendet werden können. Außerdem ist die Datei nach Ablauf der Funktion sortiert, d. h. alle Indizes sind sequentiell lesbar.

Es werden die Dateien SFILE0 und SFILE1 benötigt.

Mögliche Fehler: 1, 3, 4, 65, 68, 72

2.8.8.3. Invertieren einer Satzdatei

Aufruf in PASCAL:

```
KEYINVERT (u, f, rec, rec-element, w)
```

u	=	unit/Laufwerk Satzdatei
f	=	filename Satzdatei
rec	=	Satzstruktur der Satzdatei
rec-element	=	neuer Index
w	=	Arbeitsnummer

Es ist die Zieldatei (Indexdatei) durch OPENDIRECT zu öffnen. Die sich ergebende Arbeitsnummer wird im Aufruf KEYINVERT als 5. Parameter angegeben.

Die durch die Arbeitsnummer (w) angegebene Indexdatei wird mit dem Teil der Sätze aus der Datei (u, f) gefüllt, der durch den 4. Parameter (rec-element) bestimmt ist. Der 4. Parameter muß Bestandteil des 3. Parameters (rec) sein (s.a. Beispiel PDFS06).

Die Indexdatei, die sich hinter der Arbeitsnummer (w) verbirgt, ist nach dem Aufruf sortiert. Wird demzufolge die Satzdatei (u, f) verkettet mit dieser Indexdatei geöffnet, so sind alle Sätze der Datei logisch nach dem 4. Parameter des Aufrufs sortiert und können indexsequentiell verarbeitet werden.

!! ACHTUNG !!

Die Freiverwaltung der invertierten Satzdatei ist durch diesen Aufruf nicht vorbereitet. Ist die Satzdatei durch einfaches sequentielles Beschreiben (ohne ENTERKEY) gefüllt worden, so steht der Freizeiger noch auf Satz 0. Nur die Funktionen ENTERKEY bzw. SORKEY erhöhen den Freizeiger. Wenn man nun weitere Indizes mit ENTERKEY eintragen bzw. danach Sätze über die Freiverwaltung eingeben will, so würde zwangsläufig der erste Satz (Nr. 0) angewählt und der Fehler 101 erzeugt. Durch die Funktion FILEREORG kann der Freizeigerauf den korrekten Stand gesetzt werden.

Mögliche Fehler: 1, 3, 4, 65, 68, 72, 102

2.8.8.4. Reorganisation einer Satzdatei/Indexdatei

Aufruf in PASCAL:

```
FILEREORG (u1, f1, u2, f2)
```

u1 = unit/Laufwerk
Satz/Indexdatei
f1 = filename/Dateiname
Satz/Indexdatei
u2 = unit/Laufwerk
Hilfsdatei
f2 = filename/Dateiname
Hilfsdatei

Die durch den 1. und 2. Parameter (u1, f1) angegebene Satzdatei wird reorganisiert.
Alle Sätze mit Satzlänge 0 (durch die Funktion DELETE gelöscht) werden durch nachfolgende Sätze belegt, so daß eine physikalische Folge ohne Lücken entsteht.
Nach Ablauf der Funktion wird der Freizeiger der Satzdatei auf der ersten freien Satz gesetzt, so daß nun über ENTERKEY und WRITES Indizes und Sätze eingetragen werden können.

Da nach der Reorganisation der Satzdatei alle Referenzen (Satznummern) der zugehörigen Indexdateien falsch sind, werden während der Reorganisation alle Platzveränderungen in der Hilfsdatei (u2, f2) protokolliert.

Um nun alle Indexdateien auf einen korrekten Stand zu bringen, muß die Funktion je Indexdatei einmal mit dem Namen der Indexdatei als 2. Parameter aufgerufen werden.
Sind alle Indexdateien angepaßt, kann die Hilfsdatei (f2) gelöscht werden (KILL,u,f).
Die Hilfsdatei wird von dem Operator FILEREORG selbständig angelegt.

Es werden die Dateien SFILE0 und SFILE1 benötigt.

Mögliche Fehler: 1, 3, 4, 65, 68, 69, 72

2.9. Fehlerbehandlung

2.9.1. Erläuterungen

Da in Standard PASCAL keine Möglichkeit einer systemunterstützten Fehlerbehandlung (z.B. bei BASIC : ON ERR) vorgesehen ist, hat man bei dieser Implementierung eine Systemfunktion vorgesehen (DFMSERROR).
Die Systemfunktion ist von ihrer Art her eine Systemvariable.
Die Systemvariable muß direkt nach einem Aufruf, der zu einem Fehler führen könnte, abgefragt werden. Bei einem erneuten Aufruf einer DFMS-Operation wird die Systemvariable DFMSERROR auf 0 gesetzt wird. Die Abfrage kann in einem CASE Statement oder in einem IF Statement erfolgen. Es besteht keine Einschränkung gegenüber einer Benutzerfunktion. Die Funktion ist vom Typ INTEGER.

2.9.2. Beispiel

```
32  (*****************************************************)
33
34  PROCEDURE FEHLER(TEXT:STRING15);
35  BEGIN
36  WRITELN('FEHLER : ',DFMSERROR:4,' BEI ',TEXT)
37  END;
38  (*****************************************************)
39


70  BEGIN
71  READNEXT(ARB1,SATZ);
72  CASE DFMSERROR OF
73       0:   DISPLAY(SATZ);
74     100:   WRITELN(' ENDE DER DATEI ');
75     101:   BEGIN
76            WRITELN(' SATZ IST LEER ');
77            NEXT(ARB1)
78            END;
79  OTHERS:   FEHLER('SATZ-SEQUENT.  ')
80  END
81  END;
82
```

2.9.3. Fehlerliste

1	Gerät defekt oder nicht in der Geräteliste eingetragen - eintragen mit SERV,MOUNT -.
3	Gerät nicht betriebsbereit
4	Am Plattenlaufwerk ist der Hardware-Schreibschutz eingeschaltet
5	Lesefehler (CRC) beim Zugriff zum Plattenspeicher.
65	Datei bei CREATE oder CRIND schon vorhanden, bzw. bei Lese- und Änderungsaufrufen nicht vorhanden.
68	Verlangte Zugriffsart nicht erlaubt
69	Plattenüberlauf bei CREATE oder CRIND.
72	Falscher Dateityp
100	Ende der Datei, bei unsortierten Indexdateien schon nach dem ersten Eintrag.
101	Der beschriebene Anteil des anghewählten Satzes ist kürzer als die aufnehmende Variable READS oder READNEXT statement bzw. ist der noch beschreibbare Anteil des angewählten Satzes kürzer als die abgebende Variable im WRITES oder WRITENEXT Statement.
102	Verlangte Zugriffsart stimmt nicht mit dem Typ der geöffneten Datei überein oder Arbeitsnummernvariable enthält keinen signifikanten Wert.
103	Index schon vorhanden: Bei Indexdateityp 32 oder 96 (doppelte nicht zugelassen). Index nicht gefunden: Bei Suchaufträgen
104	Parameter eines DFMS-Aufrufes haben einen unzulässingen Wert.
105	Opentabellenüberlauf oder Arbeitsnummer hat falschen Wert.

3. Anhang

3.1. Verwendete Abkürzungen

Sachwort	Erläuterung, Seite etc.
!Cu	S. 7, transiente Datei, wird vom System angelegt und gelöscht (u = user/Benutzer).
<cr>	S. 7, Tastenfunktion Carriage Return/Wagenrücklauf Abschluß eines Kommandos.
ANSZ	S.12, aktueller Parameter bei der Prozedur CREATE; gibt die Anzahl der Sätze an.
ASCII	American Standard Code for Information Interchange.
CHR	S. 52, Wandlungsfunktion von numerisch nach ASCII
COMSYS	S. 7, Bediensystem für alle Übersetzer; erzeugt eine einheitliche Schnittstelle.
CPU	S. 4, Central-Processor-Unit Zentraleinheit eines Rechnersystems.
CRIND	S. 27, Create Index, Erzeugung einer Indexdatei.
DFMS	S. 5, Data File Management System verwaltet satzstrucktuierte Dateien.
DIAL	S. 16, Meldung des BS auf dem Benutzerterminal
E/A	S. 4, Ein - Ausgabe
EOF	S. 14, Ende der Datei; End of File.

Sachwort	Erläuterung, Seite etc.
EOR	S. 26, Füllhöhenzeiger; End of Record.
FZ	S. 26, Freizeiger; wird im Beschreibungsvektor einer Satzdatei geführt.
f	filename/Dateiname; kann 1 bis 6 Zeichen lang sein.
fa	filename/Dateiname alt
fn	filename/Dateiname neu
GETKNEXT	S. 45, Lesen des aktuellen Indizes und Fortschaltung.
ISAM	S. 3, Indirekter Zugriff über einen Ordnungsbegriff; Index-Sequentiel-Access-Method.
KB	S. 5, siehe KByte.
KByte	S. 4, 1024 Byte, bezeichnet Hauptspeichergrößen oder Diskettenvolumen.
KEYSORT	S. 28, Sortieren von Indizes für logisch sequentielle Zugriffe.
MByte	S. 4, 1048576 Byte; zur Bezeichnung von Platten- oder Hauptspeicherbereichen.
MEM	S. 7, Abkürzung des Kommandos MEMORY; es wird die Größe der Partition ausgegeben.

Sachwort	Erläuterung, Seite etc.
MODNEXT	S. 40, Modifizieren eines Satzes (Überschreiben) mit Freigabe des Satzes.
OBJ,u,f	S. 24, Kommando des Bediensystems zur Erzeugung einer Objektdatei.
PDFS04	S. 46, Beispielprogramm Pascal Data File Sytsem und laufende Nummer.
PROT	S! 19, Kürzel für die Funktion PROTECTION (Schutzkode) setzen.
p	S. 20, Protection/Schutzkode
rec	S. 24, record/Satz Festlegung der Satzstruktur
READNEXT	S. 36, Lesen des aktuellen Satzes mit sequentieller Fortschaltung.
READS	S. 36, Readsentence/Lese Satz ohne sequentielle Fortschaltung
SEKEY	S. 46, Suchen eines Indizes und setzen des Satzzeigers; Selectkey.
SELDIRECT	S. 43, Selectdirect; direkte Anwahl eines Satzes über Vorgabe einer Satznummer.
SELINDEXED	S. 43, Indirekte Anwahl durch Suchen eines Indizes und Setzen des Satzzeigers.
SORKEY	S. 51, sortiertes Eintragen eines Indizes; auch wenn die Datei vom Type unsortiert ist.

Sachwort	Erläuterung, Seite etc.
SORKNUM	S. 51, sortiertes Eintragen wie bei SORKEY; allerdings mit Vorgabe der Satznummer.
snr	S. 43, Satznummer
TSOS	S. 5, Time-Sharing-Operating-System.
t	S. 27, Type einer Indexdatei; sortiert, unsortiert, doppelte Indizes zugelassen oder nicht.
u	unit/Laufwerk
w	S. 31, worknumber/Arbeitsnummer; wird vom Sytem vergeben.
WRITENEXT	S. 37, Beschreiben des aktuellen Satzes mit sequentieller Fortschaltung.
WRITES	S. 37, Writesentence; Satz schreiben , ohne sequentielle Fortschaltung.
XOSPRO	S.12, Systemprolog; enthält Deklarationen von Prozeduren und Funktionen.

3.2. Glossar

Sachwort	Erläuterung, Seite etc.
Abhängiges Ändern	trifft dann zu, wenn der neue Inhalt eines Satzes vom alten Inhalt abhängig ist.
Aktuelle Parameter	S. 12, Variable -beim DFMS-, die Werte beim Prozeduraufruf an den Modul übertragen werden.
Aktueller Autorkode	S. 16, kann durch ein Kommando gesetzt, verändert und gelöscht werden.
Aktueller Index	S. 52, derjenige Index, auf den der Satzzeiger (SZ) zeigt.
Aktueller Satz	S. 36, ist der Satz, auf den der Satzzeiger (SZ) zeigt.
Arbeitsnummer	S. 31, wird vom System vergeben und bezeichnet einen Platz in der Opentabelle.
Autorkode	S. 16, man unterscheidet den "aktuellen Autorkode" und Datei-Autorkode.
Beschreibungsvektor	S. 26/27, insgesamt 256 Byte Vorspann einer DFMS-Datei; enthält Daten über die Datei
Blockverkettung	Verkettung eines Blocks mit den Indizes im Indirektbereich.
Call by reference	S. 12, Übertragungsart; um Werte an eine Prozedur/Funktion und zurück zu geben.

Sachwort	Erläuterung, Seite etc.
Freizeiger	S. 26, wird im Beschreibungsvektor einer DFMS-Satzdatei bei ENTERKEY geführt.
Füllhöhenangabe	S. 26, siehe End of Record.
Funktionsaufruf	S. 14, System u. Benutzerfehler werden durch die Funktion "DFMSERROR" gemeldet.
HASH-Zeiger	S. 29, auch Suchverkettung genannt. Verkettet alle Indizes mit demselben Hash-Wert.
Hilfsroutinen	S. 55, Hilfsroutinen sind in der Biliothek untergebracht und im Overlay zu nutzen.
INITPROCEDURE	S. 13, Möglichkeit bei dieser Imlementierung Variable vorzubesetzen.
Implizite Satzanwahl	S. 46, wird durch die Angabe eines Indizes (auch mit Maskenzeichen) erreicht.
Index-sequentiell	Zugriffsmöglichkeit zu Sätzen und Indizes, auch logisch sequentiell genannt.
Indexanwahl	S. 48, es kann der kleinste Index angewählt werden. Zustand nach dem Öffnen.
Indirektbereich	S. 29, Indizes, die nicht mehr in den Direktbereich passen, werden dort eingetragen.
Indirekte Satzanwahl	S. 44, durch Anwahl eines Indizes und Setzen des Satzzeigers auf die Referenzangabe.

Sachwort	Erläuterung, Seite etc.
Inhaltsverzeichnis	S. 15auch Directory genannt, enthält alle Dateien mit Informationen von einer Platte.
Invertieren	S. 58, Aufbauen einer Indexdatei anhand des Inhaltes eines Satzes.
Maskenzeichen	zeigen nicht bekannte Teile eines Suchbegriffs an. Es werden Sternchen "*" verwendet.
Öffnen einer Datei	S. 35, beim Öffnen werden die zugriffsrelevanten Daten aus dem Inhaltsverzeichnis gelesen.
Opentabelle	S. 30, nimmt die relevanten Informationen einer Datei beim Öffnen der Datei auf.
Ordnungsbegriff	auch Schlüssel oder Index genannt.
Overlaytechnik	Überlagerungstechnik; immer dann notwendig, wenn der Operator nicht reentrant ist.
Partition	S. 5, Hauptspeicherbereich, der einem Programm (einem Benutzer) zur Verfügung steht.
Permanente Reorg.	S. 23, Reorganisation des Plattenspeichers, die vom System autom. vorgenommen wird.
Prolog	Deklarationsteil, der jedem Pascal Quellprogramm als Vorspann dient (s.a.S. 2)
Prozeduraufruf	S. 12, alle Funktionen und Operatorn des Moduls DFMS werden über Prozeduren aufgerufen.

Sachwort	Erläuterung, Seite etc.
Reentrant	wird ein Programmstück genannt, wenn es parallel nutzbar ist.
Referenzen	S. 61, Satznummern innerhalb einer Indexdatei, die den indirekten Zugriff ermöglichen.
Reorganisieren	S. 57, eleminieren der Freibereiche in einer Index- oder Satzdatei.
SFILE0	S. 55, Systemdatei, wird beim Sortieren, Reorganisieren und Invertieren benötigt.
SFILE1	S. 55, siehe SFILE0.
Satzzeiger	kann explizit vom Programm bzw implizit durch andere Aufrufe verändert werden.
Schliessen Datei	S. 35, beim Schliessen werden die benutzten Plätze der Opentabelle freigegeben.
Schutz-Kode	S. 18, wird vom System beim Anlegen einer Datei auf '34 gesetzt (private Datendatei).
Segment	S. 56, Teil eines Programms, der zur Laufzeit vom Plattenspeicher nachgeladen wird.
Sequentiel Access	S. 3, sequentieller Zugriff; über Indizes oder Satznummern möglich.
Seriell reusable	S. 55, ein so bezeichnetes Programmstück ist nicht parallel nutzbar.

Sachwort	Erläuterung, Seite etc.
Sortieren	S. 56, setzen der Sortierzeiger, sodaß alle Indizes logisch sequentiell abrufbar sind.
Speicherbelegung	S. 5, wird bei der Generierung eines BS ausgegeben.
Sperren eines Satzes	S. 38, wird für das Aktualisieren eines Satzes benötigt.
Sternchen	dienen als Maskenzeichen beim Suchen eines nur teilweise bekannten Indizes.
Syntaxdiagramme	S. 12, Hilfsmittel, um Sprachelemente einer Programmiersprache eindeutig zu beschreiben.
System-Prolog	S. 12, siehe Prolog
Systemfunktion	S. 14, im Prolog definiert; "DFMSERROR", enthält eventuelle Fehler nach einem Aufruf.
Systemplatte	S. 5, auch Systemresidenz genannt.
Systemresidenz	das Plattenlaufwerk, von dem das BS oder das Sprachsystem geladen wurde.
Systemumfeld	S. 7, Hard- und Softwarekomponenten, die für einwandfreie Arbeit notwendig sind.

Sachwort	Erläuterung, Seite etc.
Systemvariable	S. 62, stellt aufgrund von sich ändernden Systemzuständen einen neuen Wert zur Verfügung.
Transiente Datei	S. 7, wird vom Bediensystem "COMSYS" mit dem Namen !Cu angelegt (u=user/Benutzernr.).
Vergleichsoperator	S. 46, dient zum Anwählen des nächstgrößeren oder nächstkleineren Indizes.
Verkettete Datei	S. 36, besteht aus einer Satz- und Indexdatei. Der Zusammenschluß erfolgt beim Öffnen.
Verkettetes Öffnen	S. 33, öffnet zwei Dateien, und belegt zwei Pläzte in der Opentabelle.
Verkettung	S. 28, Vor- und Rückwärtsverkettung für die Sortierzeiger und Blockverkettung.
Wahlfreier Zugriff	S. 3, auch Random-Zugriff genannt; sowohl über Index als auch über Satznummer möglich.
Zugriff indexabsolut	S. 33,indirekter Zugriff zu einem Satz oder direkter Zuggriff zu einem Ordnungsbegriff.
Zugriffmechanismen	S. 28, auch Zugriffsmöglichkeiten genannt.
Zugriffswege	S. 3, Random über Satznummer, Random über Index, Index-und satzsequentiell.

Heinrich Dietz
Solinger Straße 9
4330 Mülheim-Ruhr
Tel.: (0208) 4434-1
Telex 856770

DIETZ Computer SYSTEME