2. PRINCIPAL ASPECTS OF DATA MANAGEMENT

This chapter describes the characteristics of data and files as seen by the system. Although much of this will be familiar to the experienced programmer he should nevertheless read this chapter before using the rest of the manual in order to understand how the terminology and concepts of data and files are used on the PTS system.

2.1 Files

When using the word *file* it is assumed that the data collected in a file is recorded in such a way that it can be read by a machine. This restriction places a file within the field of automatic data processing, within the limits of a computer program.
Thus, a file is a machine-accessible collection of data which should be organized logically according to the accessibility requirements of the separate data elements and the overall collection.

A file consists of a number of *records*, each record giving information on a specific subject. In an account file each record describes one account. A record consists of a number of data items. The current balance, for example, is one of the data items in an account record. In a program the data items of a record can be given names so that they can be referred to and processed.

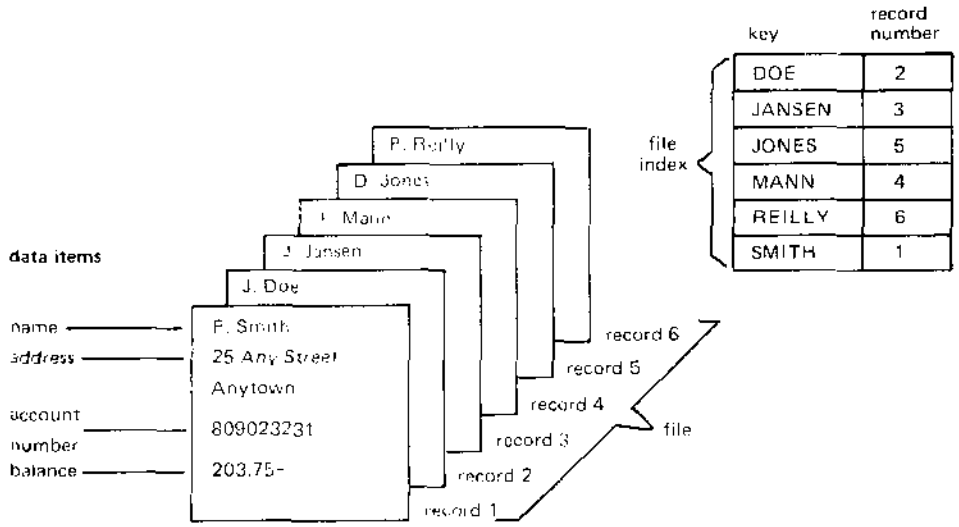


Figure 1.1 Composition of a file (with an optional index)

2.1.1 *Use of data files*

Data processing, at least in business applications, is very often a matter of routine. Files give a permanent description of some aspect of the business environment. As long as the information system manages to keep this description up-to-date, efficient processing may be done on the basis of this description.

Some processing only uses data stored in permanent files. If a user wants to know the current balance of an account holder, it is sufficient to select the amount from the relevant file, assuming that the file has been kept up-to-date. Keeping files up-to-date implies that changes must be made as a result of some calculation (an amount paid in is added to the account balance), or to some permanent item (the account holder's address).

Thus two types of data may be distinguished:
- input data
- permanent data (note: the existence of such a data item is permanent; the value of the item may be changed).

The new data could be input to the system in one of two ways:
- it could be gathered into input files from one or more sources and presented to the system as a batch of changes to be processed at one time (batch processing).
- it could be input via a terminal during the actual transaction that generates the change (on-line transaction processing).

2.1.2 *Retrieving data*

When processsing a file, data will be retrieved from the file, i.e. specific units of data are separated from the file and considered by the processing system. If necessary, the data is modified and rewritten to the file. For instance, when updating an account record it will be retrieved from the file and, after modification, be rewritten. Sometimes several data items of the same record must be considered and if necessary, updated at the same time, e.g. a woman account holder gets married and changes her surname and address.

When serving a customer, his account record will be required. Since the correct account record must be located, the record must have some identification in the form of a unique name or number which can be reffered to during the transaction.
For this purpose "keys" are used. When the key is known, the corresponding record can be retrieved from the file. The account number could be the key to the customer's account (this item is also a data item of the record required). Such an item is called the record key. Another example of a record key is the account holder's name.
The kind of key used is important when considering the methods of accessing the record in the file and possibly the organisation of the file.

In the case of a numeric data item such as the account number, the file can be constructed in such an order as to ensure that the last few digits of the number represent the record's actual position in the file. Account number 80803237 could be the 3237th record relative to the beginning of the file. This record key is called the 'logical record number'.
However, if the file is ordered according to the alphabetic order of account holders' names some other kind of key must be used because there can be no numerical relationship between alphabetic data items. The record key must be cross-referenced by an index to give the logical position of the record on the file.
Thus it can be seen that two kinds of record key exist:
- the logical record number that permits direct access to the record
- an index that provides a cross-reference to the record on the disk.

2.1.3 *Accessibility of Files*

The main mode of operation of the PTS 6800 terminal system is 'on-line processing'. This means that most accesses to files will be as a direct result of a request from one of the work positions. One work position should be servicing on account holder's transaction (deposit or withdrawal of cash) and another work position could want reports on the current status of account. In this mode of operation it is probable that more than one task wants access to the same file at the same time. Each of these tasks can treat the data in an entirely different way.
Consider the following file composed of 10 accounts.

1	2	3	4	5	6	7	8	9	10

Task A0 wants only the data relevant to the present customer. Customers come into the office at random, so the data accesses would be random, for example the following sequence might occur:

7, 5, 8, 3, 4, 6, 2 . . . . . .

Task B0 wants a report on the present balance of each account, so would access the file sequentially in account number order:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Task C0 wants the present status of certain accounts accessed with the account holder's name. In this example, the system must have a cross-reference index between the name and the account record. Accesses for task C would appear thus:

F. Smith	J. Doe	J. Jansen	A. Mann	– index
↓	↓	↓	↓	
8	7	9	3	– record

The access is random (the order of requirements cannot be predetermined) and via an index.

Physically the data file is the same for all tasks. The system itself only sees one kind of file. However, each task sees a different logical relationship between records and as a file is organised according to the logical relationship between records, each task 'sees' a different file organisation.

It is important to note the distinction between access methods and types of file. There are only two methods of accessing a record:
– direct access (either the 'next' record from the current position or via a logical record number)
  indirect access (via an index).

File organisation is a logical concept that describes the relationship between record accesses as seen by the task. There are three kinds of logical file:
– sequential (as seen by task B0)
– random (as seen by task A0)
– indexed random (as seen by task C0)

Each kind is described in more detail in section 2.1.7 and chapters 4, 5 and 6.

2.1.4 *File turnover and growth*

During the normal course of business, some accounts are closed and some are opened and the corresponding data in the accounts will be changed. Data is deleted and new data is inserted. These activities are called 'file turnover'. There may be no more data in the file than there was before. The file did not grow but part of it was deleted and somewhere else a comparable set of data was inserted.
File turnover places certain requirements on the construction of a file. For example, new records cannot be inserted in the free space left by the old records, although items within the record can be replaced by new values. Different tasks can use different rules to determine the validity of new data. A file of accounts organized by account holder's name could have problems if there are two account holders by the same name (like father and son, or coincidences with common names).

Associated with the problems of file turnover is that of file growth. The number of records must be specified at the time the file is created. Allowance must be made for an increase in the user's business by specifying enough empty records for the file to grow without needing re-organizing. The factors to be balanced are:
– availability of disk space for 'empty records'
– rate of growth of the file (especially in the initial set up of the installation when data is being transferred onto the system)

– time available for re-organising files (busy installations cannot afford to waste time on frequent 'housekeeping' jobs).
– nuisance factor to the operator (if files are constructed without enough room for growth, they may cause overflows requiring frequent re-organization).

2.1.5 *File Organization*

The PTS system recognizes three kinds of file organization:
– standard files (type S)
– library files (type L)
– non-standard files (type X)

A standard file is one that has been formatted under the TOSS system for use by an application that operates under TOSS. All the files described in this manual, are 'standard files' and all these files are understood to contain data, or index records.

A library file is one that contains program coding in one form or another, i.e. as source, intermediate object or loadable form. This kind of file is outside of the scope of this manual and the reader should refer to manuel M11, the DOS System Reference Manual, or M08, the TOSS Utilities Reference Manual.

A non-standard file is one that is either unformatted, or from a computer system that uses different labelling and formatting standards. This kind of file is outside the scope of this manual and the reader should refer to manual M11, the DOS System Reference Manual, or M08, the TOSS Utilities Reference Manual, in order to process this kind of file.

2.1.6 *File categories*

All files used by the Data Management instructions can be divided into the following 'file categories':
a. *Data files* – these contain information that is processed by the tasks and could contain records about accounts, account holder's etc. A data file on its own can be used for sequential or random requests.
b. *Index files* – these contain a list of symbolic keys that are used to reference records in data files. The use of an index file considerably reduces search time for a record, especially if the record can be referenced by more than one key. Index files can be treated as a data file for updating purposes (sequential requests) or used as the index to a data file (indexed random requests).
c. *Master index files* – this is a 'summary' of the index file that is produced after the index file has been created. It reduces the time required to search an index file and is used by the system in conjunction with indexed random requests. The master index file is held in memory after its relevant index file has been assigned to a task.

With indexed requests, these different files are related by pointers. A data file can be associated with more than one set of index/master index files but these latter cannot exist without a data file. A set of data, index and master files constitute a 'file structure'. A file structure can only contain *one* data file.

2.1.7 *Data File Organization*

Data files can be organized in one of three ways:
a. *sequential* – the file is created and accessed in such a way that records are processed serially.
b. *random* – there is no relationship between records and they are required randomly. Each record is accessed by its position relative to the beginning of the file via its logical record number.
c. *indexed random* – records are accessed via a key that is contained in an 'index file'.

The kind of organization used depends basically on the use of the file. A sequential file is used where actions are always carried out in a sequential order, for example list processing, reporting on the state of accounts, logging various activities on-line as they occur. Sequential files are more often seen in batch processing, but could be used for on-line processing in some circumstances, for example log files, see chapter 3.2.
A random file is used when the accesses are happening at random times and to randomly required records. This kind of file organization is more often seen in on-line processing where the accesses are coming from any number of terminals dealing with randomly occurring events, for example, customers walking into a bank to deposit or withdraw cash from their accounts. It would be nonsense to expect the customers to visit the back in alphabetical order so the files must be organised to allow records to be accessed directly — one of the data items in the record must be used to indicate the record's position in the file relative to the start of the file.
The usefulness of a random file is limited if records are to be accessed by a choice of items, for example, an account file could be accessed by either the account number or the account-holder's name. If this is the case then all the items used for access (the keys) must be set up in a cross-reference file (the index) that gives the logical record number related to all the keys. This method of organization, a data file with an index, is called indexed random.

Each kind of data file organization is described in the following chapters.

2.2 Volume Organization

A volume is a single physical unit capable of holding information.
For the purpose of this manual this is understood to be:
– a removable disk cartridge
– a fixed disk
– a flexible disk

2.2.1 *Disk Structure*

Each disk volume is divided into cylinders, each cylinder into tracks, and each track into sectors. The user program does not use this structure as it only addresses records within a file. The programmer must be aware of this structure when constructing files as it could affect the blocking factor of records within blocks, number of file extents in the volume, or number of volumes required for one large file. The structure of the PTS 6875/76 disk is shown in the figure below.

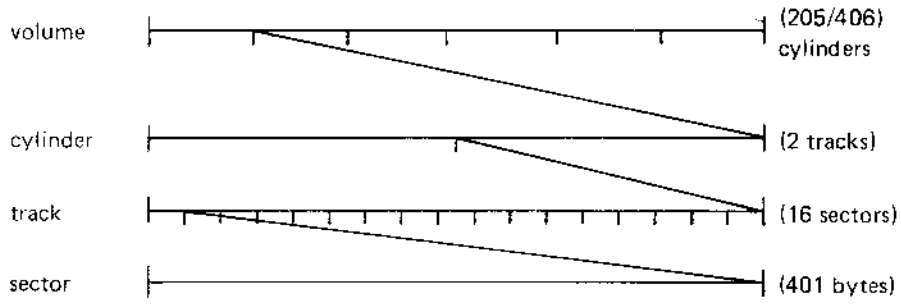


Figure 1.3 Disk Structure, PTS 6875/76

Each sector can be subdivided into records according to the program's requirements. The number of records stored in each sector is called the blocking factor and (record size X the blocking factor) should never exceed 401 bytes. The largest record allowed is 400 bytes + 1 status byte used by the system. Every record in the file has one byte reserved for system purposes so a record of 80 data bytes must be created as a record with the length 81. When accessing the record, the program uses the length '80'. Thus only four records could be blocked onto one sector (5 x 81 = 405, is too large). If the record length could be reduced to 79 data bytes + 1 status byte then five records could be blocked onto one sector, making more efficient use of disk space.

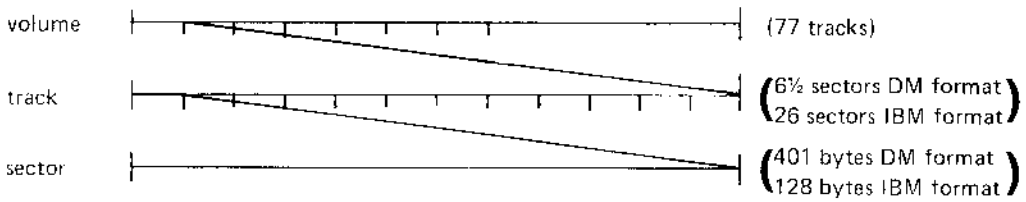


Figure 1.4 Flexible Disk Structure

disk model	*Cylinders per volume*	*Tracks per cylinder*	*Sectors per track*	*bytes per sector available to the user*
PT8 2 x 2¼ M	203	2	16	401
PTS 6876 2 x 5 M	406	2	16	401
		tracks per volume		
PTS 6879 (flexible disk)	—	77	6½ 26	401 DM format 128 *IBM format*

Table 1.1 Disk Capacity Available to User Programs

Some of the space on both disks and flexible disks is reserved for system use and so can never be accessed by user programs, see the table below.

disk model	*system-reserved areas*	*Purpose*
PTS 6875 PTS 6876	cylinder 00, track 00, Sector 00	Volume Label
PTS 6876	cylinder 00, track 00, Sector 01	Initial Program Loader (IPL)
PTS 6875	cylinder 200 203	system use
PTS 6876	cylinder 406 407	system use
flexible disk (PTS 6879)	track 00, Sectors 01-04 (128 byte sectors	Volume Label
	track 00, Sectors 05-08 (128 byte sectors	IPL

Table 2 Reserved Areas

Note that both disks and flexible disks will have variable amounts of space allocated to the Volume Table of Contents (VTOC) depending upon the number of file extents.

The VTOC contains one record of 41 bytes for every file extent on the volume. VTOC records are blocked 9 per sector so simply divide the number of file extents by 9 to find the number of reserved sectors.

The VTOC is accessible only through Assembly routines, so the reader is referred to the Assembler Programmer's Reference Manual.

2.2.2 *Creating a volume*

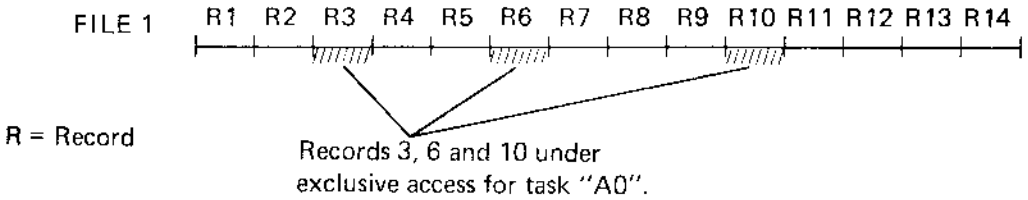
A disk volume cannot be used on the PTS6800 system until it has been initialized and formatted by the Create Volume utility (CRV). A full description is available in the Utilities Reference Manual, M08 an in M11 DOS6800 Reference Manual (TOSSUT Utility), but is mentioned briefly here. CRV writes a volume label and an empty VTOC, then writes cylinder identifiers in all sectors. This identifier is outside of the area of the sector that is available to the user. CRV also performs a quality test on each sector by writing then reading back. If any defective sectors are found CRV creates a dummy file called 'BADSPOT' and assigns all unusable sectors to that file. IPL is written to the disk and CRV terminates. The volume is then available for use, unless any badspots are located in the area reserved for the volume label on IPL.

2.3 Record handling

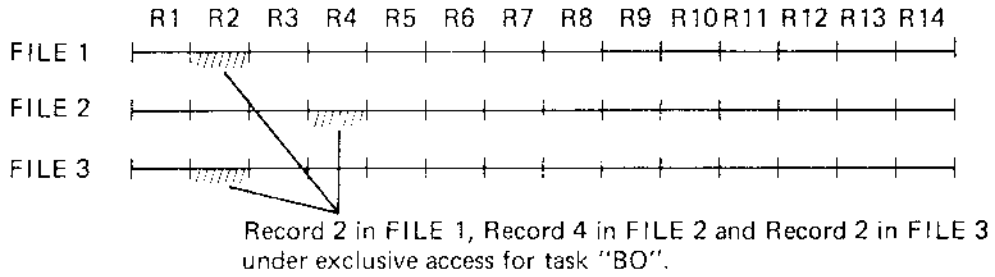
2.3.1 *Exclusive access*

Data files may be shared by a number of tasks, so simultanuous updating of records must be prevented. Exclusive access is a function which is used to prevent simultaneous updating or records. The exclusive access function for use by the user program is included at system generation time, but is superfluous when only one task exists using data management. However, the user still has the possibility to allow exclusive access setting for a record as an option in the instruction (assuming that exclusive access was included at system generation time).
Exclusive access is controlled on record level, which means that individual records can be held under exclusive access (no other task can get these records) but not the whole file.
In this way a task may have in one file more than one record under exclusive access and the task can have records under exclusive access in different files.
In one file different records can be under exclusive access for different tasks.

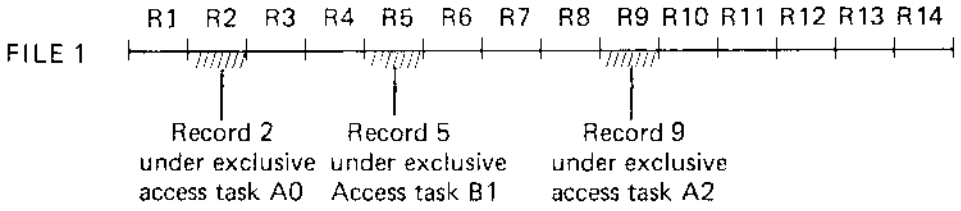
Example 1



Example 2



Example 3



Exclusive access is not used for index files, only for data files.

A record can be set under exclusive access, after an unsuccessful read operation. Exclusive access is released after a (re)write, delete or release exclusive access operation of the record has been performed.
When a record is accessed which is already under exclusive access by another task, a status is returned indicating "record protected".

2.3.2 *Current record number*

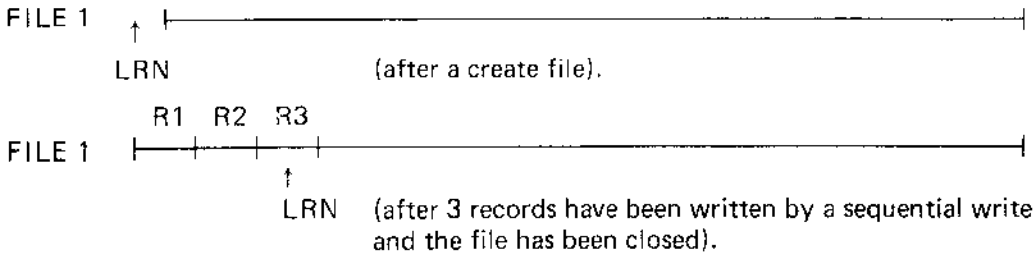
Data mangement has an internal handling of Current Record Number (CRN). For each file structure (files consisting of one data file and possibly index and master index files) an area is reserved per task in the system by data management, in which the CRN from the last request on this file is stored for each task.
Some instructions use this CRN value before execution to obtain the next record. After execution of the instructions the CRN may be updated by data management, depending on the type of instruction (see table 2.3 below). This current record number can be obtained by the user with the command 'get currency index' or 'get currency data'. In this case the last accessed record of the index and data files, respectively, is returned to the user for this task.

Instruction	*CRN used for execution of the instruction. Affected on file type:*	*CRN-updated after execution of the instruction. Affected on file type:*
Sequential Read	Data file	Data file
Sequential Write	—	—
Random Read	—	Data file
Random Write	-	Data file
Random Delete	—	—
Indexed Read	-	Data file + Index file
Indexed Rewrite		Data file
Indexed Delete	--	
Indexed Insert	—	Data file + Index file
Indexed Read Next	Index file	Data file + Index file

Table 2.3 Current record number handling

2.3.3 *Last record number*

Data management holds per file (not per task) a last record number pointer (LRN) indicating until which record the file is filled. The user cannot access this pointer, but can be informed by means of an error return code, or via the control word after a sequential write.
When a file is created by the utility Create File, the last record number pointer is always set to the beginning of that file. After a sequential write the pointer is updated every time a record is written. The sequential write and Indexed Insert instructions will influence the updating of the last record number pointer. However, the LRN is not put onto the disk until the file has been closed.



By the indexed and random instructions the user is able to read or write records located after the last record number pointer. However, an error code is returned indicating an "End of File" condition, but the I/O operation is *not* aborted. It is *up to the user* to decide whether an action has to be taken or not. When a sequential read results in an error code with the "End of file" bit set, the I/O operation *will be* aborted. Note that, when a file is being processed a difference in the value of the last record number pointer may exist between the one on disk and the one updated in memory. After the file is closed this updated last record number pointer is saved on disk.