UNIT 15 FILES

Structure **Page No.**

15.1 INTRODUCTION

In this Unit, we will discuss storage data in the computers. The data is stored in computers in the form of files. We introduce you the basic terminology related to file organisation in Sec. 15.2. In Sec. 15.3, we discuss various ways in which data is organised in files. In Sec. 15.4 to Sec 15.6, we will discuss various kinds of file organisations and their advantages and disadvantages. It will be useful if you recapitulate the units of Block 2 to refresh your knowledge of syntax related to file handling in C and also Unit 14 of this block on Tree structures.

Objectives

After studying this unit, you should be able to

- define the various terms related to files;
- describe the different ways in which data is organised in files; and
- discuss the advantages and disadvantages types of files.

15.2 TERMINOLOGY

We will now define the terms of the hierarchical structure of data collection stored in computers.

1) **Field:** It is an elementary data item characterised by its size, length and type.
   For example:
   Name : a character type of size 10
   Age: a numeric type

2) **Record:** It is a collection of related fields that can be treated as a unit from an applications point of view.
   For example:
   A university could use a student record with the fields, university enrolment no., Name Major subjects

3) **File:** Data is organised for storage in files. A file is a collection of similar, related records. It has an identifying name.
   For example: "STUDENT" could be a file consisting of student records for all the pupils in a university.

4) **Index:** An index file corresponds to a data file. It's records contain a key field and a Pointer to that record of the data file which has the same value of the key field.

Indexing will be discussed in detail later in the unit.

The data stored in files is accessed by software which can be divided into the following two categories:

1) **User Programs:** These are usually written by a Programmer to manipulate retrieved data in the manner required by the application.
2) **File Operations:** These deal with the physical movement of data in and out of files. User programs effectively use file operations through appropriate programming language syntax. The File Management system manages the independent files and acts as the software interface between the user programs and the file operations.

File operations can be categorised as-

i) CREATION of the file

ii) INSERTION of records in the file

iii) UPDATION of previously inserted records

iv) RETRIEVAL of previously inserted records

v) DELETION of records

vi) DELETION of the file.

15.3 FILE ORGANISATION

File organisation can most simply be defined as the method of storing Data record in a file and the subsequent implications on the way these records can be accessed. The factors involved in selecting a particular file organisation for uses are:

- Ease of retrieval
- Convenience of updates
- Economy of storage
- Reliability
- Security
- Integrity

Different file organisations accord the above factors differing weightages. The choice must be made depending upon the individual needs of the particular application in question.

We now introduce in brief the various commonly encountered file organisations.

1) **Sequential Files**
   Data records are stored in some specific sequence e.g. order of arrival value of key field etc. Records of a sequential file cannot be accessed at random i.e. to access the nth record, one must traverse the preceding $(n-1)$ records. Sequential files will be dealt with at length in the next section.
2) **Relative Files**
   Each data record has a fixed place in a relative file. Each record must have associated with it in integer key value that will help identify this slot. This key, therefore, will be used for insertion and retrieval of the record. Random as well as sequential access is possible. Relative files can exist only on random access devices like disks.

3) **Direct Files**
These are similar to relative files, except that the key value need not be an integer. The user can specify keys which make sense to her application.

4) **Indexed Sequential Files**
An index is added to the sequential file to provide random access. An overflow area needs to be maintained to permit insertion in sequence.

5) **Indexed Files**
In this file organisation, no sequence is imposed on the storage of records in the data file, therefore, no overflow area is needed. The index however, is maintained in strict sequence. Multiple indexes are allowed on a file to improve access.

15.4 SEQUENTIAL FILES

We will now discuss in detail the sequential file organisation as defined in Sec. 15.2. Sequential files have data records stored in a specific sequence.

A sequentially organised file may be stored on either a serial-access or a direct-access storage medium

15.4.1 Structure

To provide the "sequence" required a "key" must be defined for the data records. Usually a field whose values can uniquely identify data records is selected as the key. If a single field cannot fulfil this criterion, then a combination of fields can serve as the key. For example in a file, which keeps student records, a key could be student no.

15.4.2 Operations

1) **Insertion:** Records must be inserted at the place dictated by the sequence of the keys. As is obvious, direct insertions into the main data file would lead to frequent rebuilding of the file. This problem could be mitigated by reserving overflow areas in the file for insertions. But this leads to wastage of space and also the overflow areas may also be filled.
The common method is to use transaction logging. This works as follows:
   i) collect records for insertion in a transaction file in their order of arrival.
   ii) when population of the transactions file has ceased, sort the transaction file in the order of the key of the primary data file.
   iii) merge the two files on the basis of the key to get a new copy of the primary sequential file.

   Such insertions are usually done in a batch mode when the activity/program, which populates the transaction file, have ceased. The structure of the transaction files records will be identical to that of the primary file.

2) **Deletion:** Deletion is the reverse process of insertion. The space occupied by the record should be freed for use. Usually deletion (like-insertion) is not done immediately. The concerned record is written to a transaction file. At the time of merging the corresponding data record will be dropped from the primary data file.

3) **Updation:**Updation is a combination of insertion and deletions. The record with the new value is inserted and the earlier version deleted. This is also done using transaction files.

4) **Retrieval:** User programs will often retrieve data for viewing prior to making decisions, therefore, it is vital that this data reflects the latest state of the data if the merging activity has not yet taken place.

Retrieval is usually done for a particular value of the key field. Before return in to the user, the data record should be merged with the transaction record (if any) for that key value.

The other two operations "creation" and "deletion" of files are achieved by simple programming language statements.

15.4.3 Disadvantages

Following are some of the disadvantages of sequential file organisation:

- Updates are not easily accommodated
- By definition, random access is not possible
- All records must be structurally identical. If a new field has to be added, then every record must be rewritten to provide space for the new field.
- Continuous areas may not be possible because both the primary data file and the transaction file must be looked during merging.

15.4.4 Areas of Use

Sequential files are most frequently used in commercial batch oriented data processing where there is the concept of a master file to which details are added periodically. Examples of this are payroll applications.

E6) Describe the record structure to be used for the lending section of a library.

E7) Write a program in 'C' language to insert the following records into a file 'PERSONAL'

- Adam Bede, 47, Engineer - Silas Marner, 50, Doctor

Use a name field of size 20, age field of size 2 and profession field of size 20.

E8) Merge the following, sequenced on NO:

Transactions		Master	
No.	Name	No.	Name
6	Beta	1	Delta
4	Alpha	2	Lambda
7	Gamma	8	Phi

15.5 DIRECT FILE ORGANISATION

It offers an effective way to organise data when there, is a need to access individual records directly.

To access a record directly (or random access) a relationship is used to translate the key value into a physical address. This is called the mapping function R R.(key value)– Address

Direct files are stored on DASD (Direct Access Storage Device)

A calculation if performed on the key value to get an address. This address calculation technique is often termed as hashing. The calculation applied is called a hash function.

Here we discuss a very commonly used hash function called Division - Remainder

Division-Remainder Hashing

According to this method, key value is divided by an appropriate number, generally a prime number, and the division of remainder is used as the address for the record.

The choice of appropriate divisor may not be so simple. If it is known that the file is to contain n records, then we must, assuming that only one record can be stored a given address, have divisor n.

Also we may have a very large key space as compared to the address space. Key space refers to all the possible key values. The address space possibly may not match the actual number of key values in the file, the size of key space, therefore a one to one mapping may not be there. That is calculated address may not be unique. It is called Collision, i.e.

$$R(K1) = R(K2) \text{but} K1 \neq K2$$

Two unequal keys have been calculated to have the same address. The keys are called synonyms.

There are various approaches to handle the problem of collisions. One of these is to hash to buckets. A bucket is a space that can accommodate multiple records. A discussion on buckets and other such methods to handle collisions is out of the scope of this Unit.

15.6 INDEXED SEQUENTIAL FILE ORGANISATION

When there is need to access records sequentially by some key value and also to access records directly by the same key value, the collection of records may be organised in an effective manned called Indexes Sequential Organisation.

You must be familiar with search process for a word in a language dictionary. The data in the dictionary is stored in sequential manner. However an index is provided in terms of thumb tabs. To search for a word we do not search sequentially. We access the index that is the appropriate thumb tab, locate an approximate location for the word and then proceed to find the word sequentially.

To implement the concept of indexed sequential file organisations, we consider an approach in which the index part and data part reside on a separate file. The index file has a tree structure and data file has a sequential structure. Since the data file is sequenced, it is not necessary for the index to have an entry for each record following figure shows a sequential file with a two-level index.

Level 1 of the index holds an entry for each three-record section of the main file. The level 2 indexes level 1 in the same way.

When the new records are inserted in the data file, the sequence of records need to be preserved and also the index is accordingly updated.

Two approaches are used to implement indexes are static indexes and dynamic indexes.

As the main data file changes due to insertions and deletions, the static index contents may change but the structure does not change. In case of dynamic indexing approach, insertions and deletions in the main data file may lead to changes in the index structure.

Both dynamic and static indexing techniques are useful depending on the type of application.

15.7 SUMMARY

This Unit dealt with the methods of physically storing data in the files. The terms - fields, records and files were defined. The organisation types were introduced.

The various file organisation were discussed. Sequential File Organisation finds in use in application areas where batch processing is more common. Sequential Files are simple to use and can be stored on inexpensive media. They are suitable for applications that require direct access to only particular records of the collection. They do not provide adequate support for interactive applications.

In Direct file organisation there exists a predictable relationship between the key used and by program to identify a particular record and or programmer that record's location on secondary storage. A direct file must be stored on a direct access device. Direct files are used extensively in application areas where interactive processing is used.

An Indexed Sequential file supports both sequential access by key value and direct access to a particular record given its key value. It is implemented by building an index on top of a sequential data file that resides on a direct access storage device.

15.8 SOLUTIONS/ANSWERS

1) The following record structure could take care of the general requirements of a lending library. Member No., Member Name, Book Classification, i.e. Book Name, Author, Issue Date, Due Date.

2) No model answer is given.

3) (1) Sort Transaction file

No.	Name
4	Alpha
6	Beta
7	Gamma

(2) Merge to get

No.	Name
1	Delta
2	Lambda
4	Alpha
6	Beta
7	Gamma
8	Phi

If a sequential file on a disc is to occupy the least possible space its records must be stored continuously i.e. with no unused space between them.
In case of addition or deletion of a record, the file must be rewritten to maintain its sequential order without spaces.