



Organization of the fields and records in File structure

¹Saurabh Sharma, ²Prateek Thakral

^{1,2} Department of Computer Applications, NIT, Kurukshetra, Haryana, INDIA

Abstract: A file is a collection of related records. A record is a collection of related fields. Field is a basic unit of data organization, it is the smallest logically meaningful unit of information in a file. Each field contains a single data value: e.g. last name, or balance.

Keywords: File, file structure, fields, and records.

I. INTRODUCTION

A file is a collection of data stored in one unit, identified by a filename. It can be a document, picture, audio or video stream, data library, or other collection of data. When we build file structure, we are making it possible to make data persistent. This is, one program can create data in memory and store it in a file and another can read the file and re-create the data in its memory. The basic unit of the data is the field, which contains a single data value. Fields are organized in aggregates, either as many copies of a single field (an array) or list of different fields (a record). Record is a collection of related record [1]. The paper presented here is organized as follows. Section 2 discusses the problem with file structure while entering the fields and records in a file. Section 3 discusses the various methods to solve the problem to identify the records and fields in the file structure. Finally, conclusion is done in section 4.

II. PROBLEM WITH FILE STRUCTURE

The problem with file structure is that when fields and records are entered into a file then the identity of a file as well as the record has to be maintained. Suppose the following names and addresses are used as input to the program:

Ram #160/8 Kurukshetra

Mohan #124/6 Karnal

When we list the output file on our terminal screen, here is what we see Output:

Ram#160/8KurukshetraMohan #124/6Karnal

The program writes the information out of the file precisely as specified, as a stream of byte containing no added information, once we put all the information together as single byte stream, there is no way to get it apart again [2]. We have lost the integrity of the fundamental organization unit of our input data; these fundamental units are not the individual characters but meaningful aggregates of characters, such as "ram " " 123 karnal " . When we are working with files, we call these fundamental aggregates fields. A field is the smallest logically meaningful unit of information in a file.

III. SOLUTION

This problem can be removed if we organize the field and record in such a way so that the field can be kept separate from another field as well as record can be kept separate from another record so that the identity of the records as well as fields can be preserved. There are many methods by which the fields can be separated from another field and a record can be separated from another record.

A. FIELD ORGANIZATION:

There are many ways of adding structure to file to maintain the identity of fields. Four of the most common methods are [3]:

Method 1: fix the length of field

The field in our sample file varies in length. If we force the fields into predictable length, we can pull them back out of the file simply by counting our way to end of field. Using this kind of fixed-field length structure the output can be changed so it looks like Simple arithmetic is sufficient to let us recover the data from the original field.

Sharma	Saurabh	#163/4	Kurukshetra	Haryana
Sharma	Rajni	#132/6	Karnal	Haryana

Method 2: begin each field with a length indicator

Another way to make it possible to count the end of a field is to store the field length just ahead of the field, as shown in the figure. If the fields are not too long (less than 256 bytes), it is possible to store the length in a single byte at the start of field. We refer to these fields length-based.

06Sharma07Saurabh05#162/413 Kurukshetra 07 Haryana

Method 3: separate the fields with delimiters

We can also preserve the identity of field by separating them with delimiters. All we need to do is choose some special character or sequence of character that will not appear within a field and then insert the delimiter into the file after writing each field.

Ames|...|Stillwater|OK|74075|377-1808|#Mason|...90Eastgate|Ada|OK|74820|#...

The choice of delimiter character can be very important as it must be a character that does not get in the way of processing. In many instances white spaces characters (blank, new, tab) make excellent delimiters because they provide a clean separation between fields.

Method 4: use a “keyword = value” expression to identify filed

This option has an advantage that the others do not have. It is the first structure in which a field provides information about itself. Such self describing structure can be very useful tools for organizing files in many applications.

LASTNAME=Sharma|FIRSTNAME=Saurabh|STREET=Rana|...|ZIP=74075|PHONE=377-1808|#...

B. RECORD ORGANIZATION:

A record can be defined as a set of fields that belong together when the file is viewed in terms of a higher level of organization [3,4]. Like the notion of a field, a record is another conceptual tool. It is another level of organization that we impose on the data to preserve meaning. The identity of the records can be preserved by following methods.

Method 1. Make records a predictable number of bytes

A fixed length record file is one in which each record contains the same number of bytes. The method of organizing is analogous to the first method we have discussed for making the field recognizable. We have a fixed number of fields, each with a predetermined length, that combine to make a fixed length record. This kind of fields and record structure is illustrated in the figure.

Sharma	Saurabh	#160/8	Kurukshetra	981234566
Ram	Das	#160/8	Kurukshetra	091234567

It is important to realize, however, that fixing the number of bytes in a record does not imply that the size or number of fields in the record must be fixed. Fixed length records frequently hold a variable number of variable length fields [5]. It is also possible to mix fixed and variable length fields in a record.

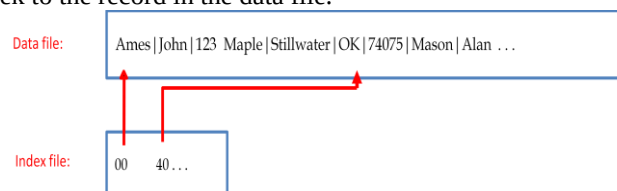
Method 2: Begin Each Record with a length indicator:

We can communicate the length of record by beginning each record with a field containing an integer that indicates how many bytes there are in the rest of the record. This is a commonly used method for handling variable-length records.

40Ames|John|123 Maple|Stillwater|OK|74075|36Mason|Alan|90 Eastgate ...

Method 3: Use an Index to Keep Track of Address:

We can use an index to keep a byte offset for each record in the original file. The byte offsets allow us to find the beginning of each successive record and compute the length of each record [6]. We look up the position of a record in the index then seek to the record in the data file.



Method 4: Place a Delimiter at the End of Each Record:

This option, at a record level, is exactly analogous to the solution we used to keep the fields distinct in the sample program we developed [7]. As with fields, the delimiter character must not get in the way of processing.

Because we often want to read files directly at our console, a common choice of a record delimiter for files that contain readable text is the end of line character.

```
Ames|John|123 Maple|Stillwater|OK|74075|#Mason|Alan|90 Eastgate|Ada|OK ...
```

IV. CONCLUSION

In this paper, discussion on various methods to preserve the identity of the fields and records has been done. There are various methods by which the identity of the fields and records can be preserved. Each method has its own advantages and disadvantages. The field can be separated from other fields by many methods like by putting delimiters after each field, delimiter can be blank space any special symbol etc which we consume 1 byte only. Another method can be using the keyword which will consume more bytes, but increase the readability of the fields in the file. So it is up to the need of the user of the file structure what method the user wants to use for field and record organization.

REFERENCES

- [1] X. Wu and A. L. N. Reddy. SCMFS: a file system for storage class memory. In Proc. of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11), pp. 1-11, 2011.
- [2] Protected and Persistent RAM File system, <http://pramfs.sourceforge.net>, 2012.
- [3] M.J. Folk and B. Zoellick. File Structures, Second Edition. Reading, MA: Addison-Wesley, 1992.
- [4] J.H. Howard et al., "Scale and Performance in a Distributed File System", ACM Transactions on Computer Systems, 6(1), Feb 1988.
- [5] T. E. Anderson, M. D. Dahlin, J. M. Neeffe, D. A. Patterson, D. S. Roselli, and R. Y. Wang. "Serverless Network File Systems", Proceedings of the 15th ACM Symposium on Operating Systems Principles, Dec 1995.
- [6] S. D. Viglas. Distributed File Structures in a Peer-to-Peer Environment. Technical report, University of Edinburgh, School of Informatics, 2006.
- [7] X. Wu and A. L. N. Reddy. SCMFS: a file system for storage class memory. In Proc. of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11), pp. 1-11, 2011.