Volume 3, Issue 12, December 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Efficient File Management Technique for Smart Card Eeprom

Shardha Porwal*, Himanshu Mittal
Department of Computer Science
JIIT University, India

Abstract— This paper examines file management issues associated with Smart card EEPROM and proposes a new technique for memory management for smart card files. The entire work concentrates to suggest a new methodology on memory allocation and de-allocation using pointers, which gives better memory utilization.

Keywords— Memory Management, Smart Card, EEPROM, Memory, File Allocation table

I. Introduction

Smart card applications were developed as an alternative to magnetic cards shortcomings [1]. The primary aim of magnetic cards is to provide machine readable data. However, the magnetic card fails in providing security against tampering. Smart Card integrates large data and provides better security on a single chip. It also provides data security against unauthorized access and any modification of data [5, 6]. Currently smart cards are used in different parts of the world for various applications including passport applications, health care applications, institute identity cards, payment systems, Telecommunication Applications, Financial Transaction and many more [9, 10]. The memory/file management issues are of great concern in today's modern applications. This work proposes a new technique for memory management for smart card EEPROM (Electrically Erasable Programmable Read Only Memory) by maintaining a list of free memory blocks to be preserve in the smart card EEPROM called the free space list. This is different from paging scheme in which whole memory is divided into number of page where a table is maintained to keep information about allocated and free pages which takes a fixed size in EEPROM memory [4].

The result of this technique has proved to be more effective in terms of memory management of EEPROM.

II. EEPROM and Smart Card Files

The EEPROM is a chip with non-volatile memory. Data and program codes can be written to and read from the EEPROM under the control of the operating system [3].

File management systems for smart cards have an object-oriented structure which means that all information about a file is stored in a file itself. Files in such object oriented systems are thus always divided into two parts. The first part, referred as the file header, contains information about the layout and structure of the file and its access conditions. The modifiable user data are stored in the second part. The file body is linked to the file header by a pointer [11]. Fig. 1 shows the internal structure of the file.

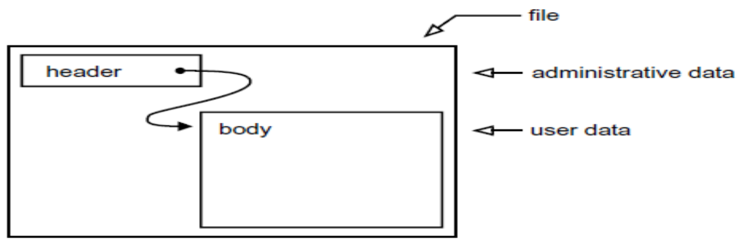

Fig. 1 Internal structure of a file for a smart card files management system

The Memory is allocated/ De-allocated for DF (Dedicated Files) and EF (Elementary Files). DF is the dedicated file just like a folders and EF is the Elementary file used to store the data [2].

III. Process for EEPROM Memory Management

The objective of this technique is to allocate /de-allocate the memory for files properly and merging the contiguous free memory blocks for further usage.

To demonstrate this technique, we have taken 20 bytes as header size for both DF and EF, the data area starting address is 1 and ending address is 3000 for simplicity.

We are maintaining a list of free blocks. The address of free block list is stored in last two bytes of EEPROM (FS_Ptr). The chain of free blocks is maintained from last bytes of EEPROM that is size of EEPROM minus the size of FS_Ptr. Fig. 2 and 3 shows the structure of nodes in Free Block List along with the step by step procedure which illustrate allocation of the memory as per our scheme.

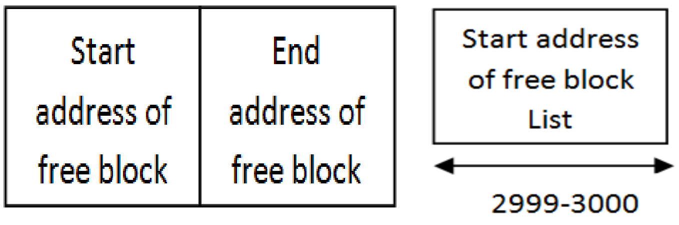
Start address of free block
End address of free block
Start address of free block List
2999-3000

Fig. 2 Structure of the node of Free Block list

Step1: Allocate the space for MF:

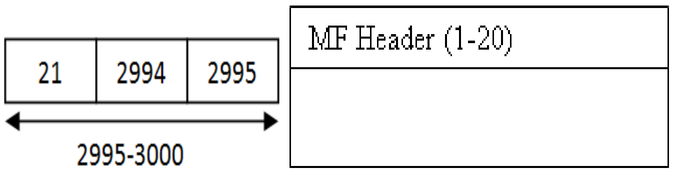

Fig. 3 (a) Allocate the memory for MF

Step2: Create 3 different DFs (allocate the memory for Header):

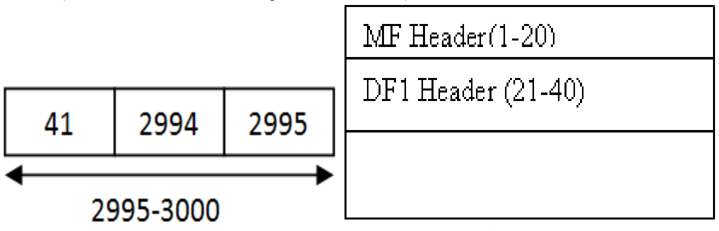

Fig. 3 (b) Allocate the memory for DF1

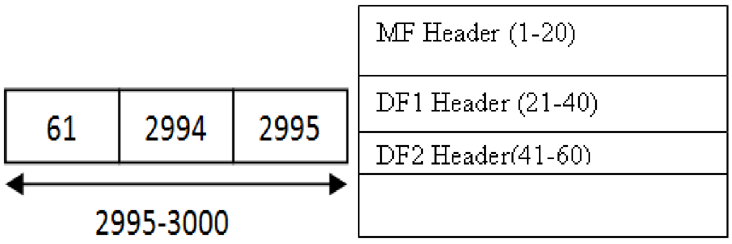

Fig. 3 (c) Allocate the memory for DF2

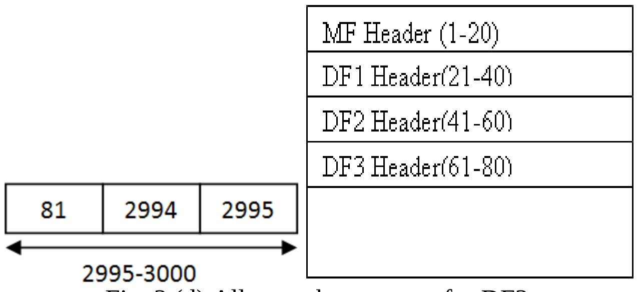

Fig. 3 (d) Allocate the memory for DF3

Step3: Now Create 2 different EFs sequentially (allocate the memory for Header, body of given size 100 bytes):

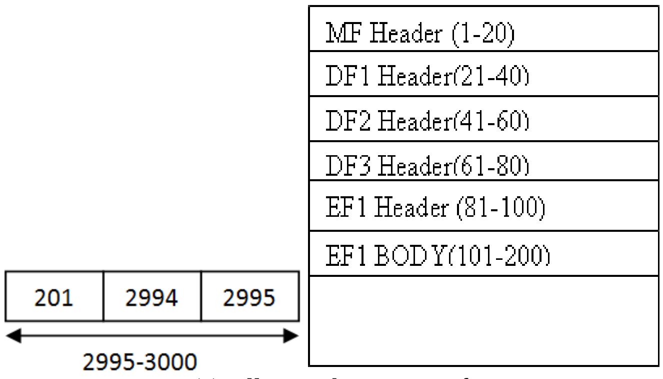

Fig. 3 (e) Allocate the memory for EF1

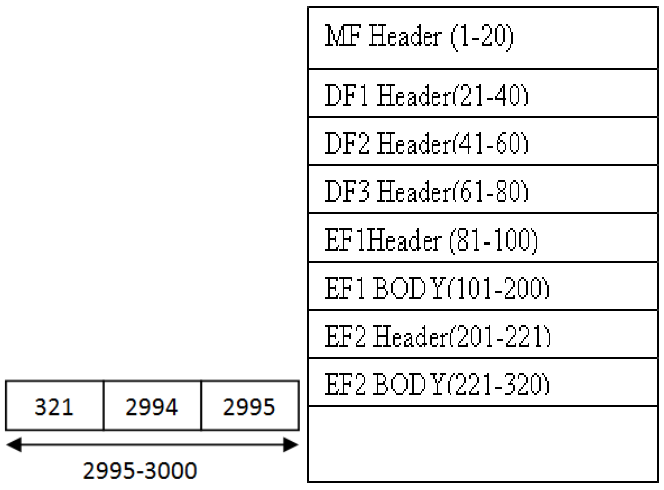

Fig. 3 (f) Allocate the memory for EF2

Step4: Delete EF1 (de-allocate the memory for Header, body of given size 100):

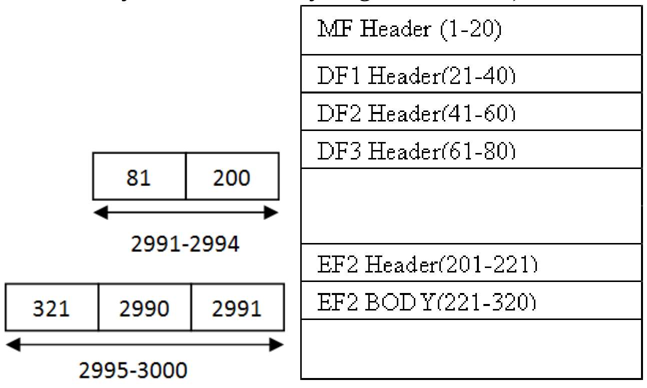

Fig. 3 (g) Free the memory occupied by EF1

Step5: Delete DF1, DF3, and DF2 (de-allocate the memory for Header, body of given size):

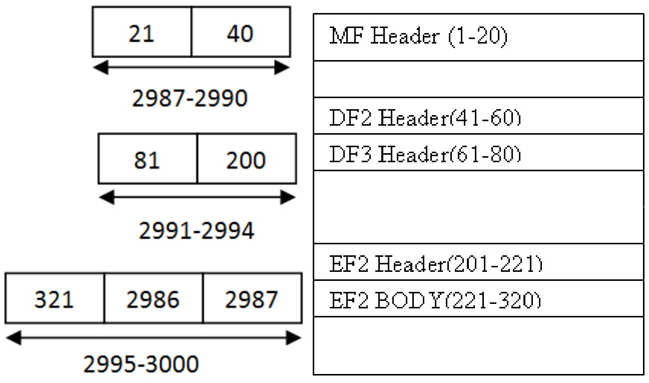

Fig. 3 (h) Free the memory occupied by DF1

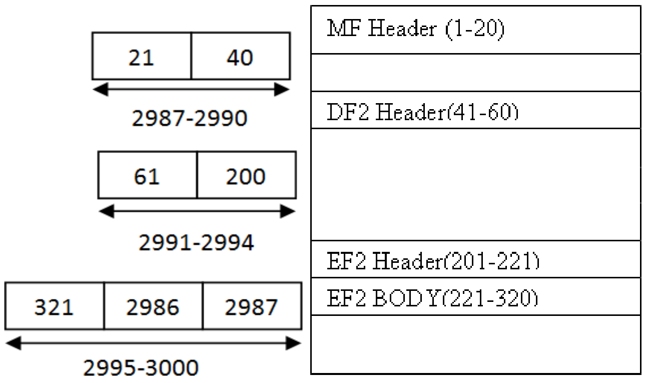

Fig. 3 (i) Free the memory occupied by DF3

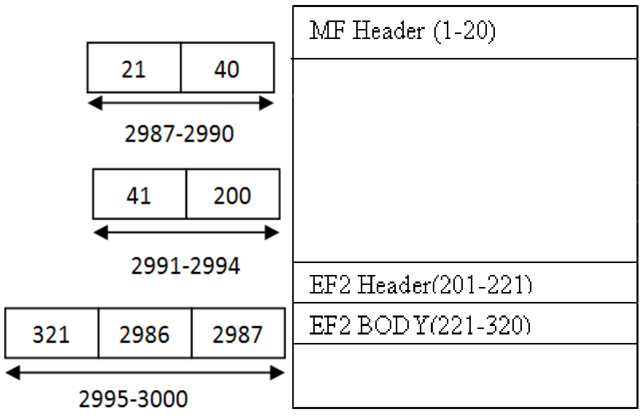

Fig. 3 (j) Free the memory occupied by DF2

Now there will be two continuous free memory blocks, therefore OS will combine them and will make a single block.

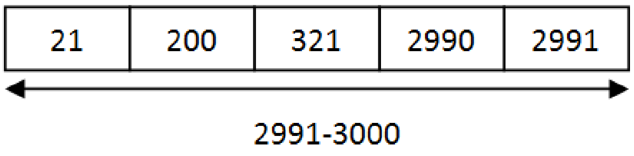

Fig. 3 (k) Combining continuous free memory blocks

Step6: Allocate memory for a DF (DF1).

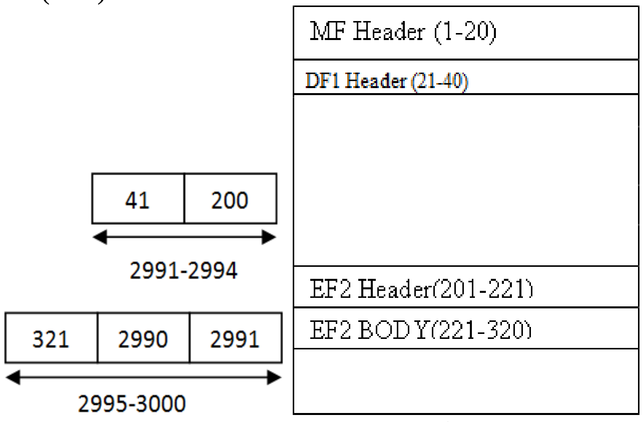

Fig. 3 (l) Allocate the memory for DF1

Step7: allocate memory for an EF3, which require 300 bytes for its body part.

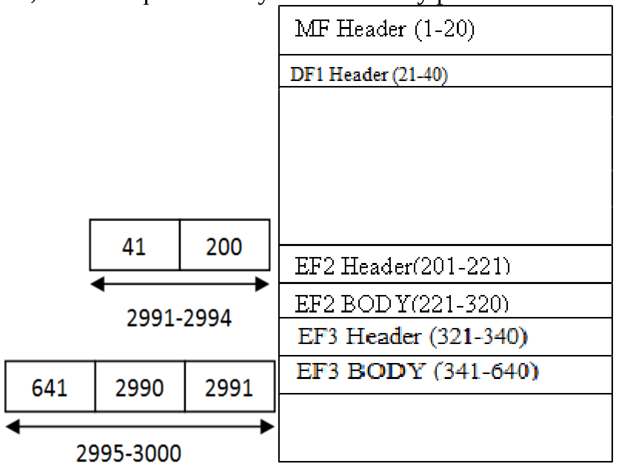

Fig. 3 (m) Allocate the memory for EF3

IV. Results and Conclusion

Memory Requirement comparison is shown for 4K EEPROM. If we divide the memory into pages and each page is of 32 bytes then numbers of pages will be 128. If we consider that page table contains page number, page status and next page fields only then it needs 128 x 3= 384 bytes, whereas our scheme requires:

TABLE I

Number of free blocks	Memory required
1	6 bytes
2	10 bytes
10	42 bytes
64 (avg.)	258 bytes

It was also observed that memory management by using the proposed scheme overcomes following issues of EEPROM such as:

1. **Easy Allocation/De-allocation:** In this, the free memory chunks are managed properly due to that reason the allocation and freeing of memory is efficient. We are maintaining only the free blocks list.
2. **No Internal Fragmentation:** The technique uses no fixed-size memory blocks, because of which there is no issue of internal fragmentation.
3. **Optimal Utilization of Memory:** The whole memory is available for storage rather than any page restriction, we can optimally use it.
4. **No Page-Size Restriction:** As the technique does not use the concept of paging so there is no need to allocate/de-allocate according to a page size.
5. **Dynamic free block List:** The free block list is updated on every request or release of a memory. The size of list varies according to the number of free blocks in memory.
6. **Accessing Time:** In paging System page table is accessed to know the status of pages while in our scheme no page table is required.
7. **Compaction:** Compaction can be done by shifting of allocated space and we get only single free block which takes little amount of memory.

ACKNOWLEDGMENT

The authors would like to acknowledge G. L. Bajaj Institute of technology and management, Gr. Noida for providing the platform for the Project on smart card OS, Prof. Naresh Maheshwari for his valuable guidance, Mr. Anil Kumar Singh, director of Amity s/w inc. for the fruitful discussion and Mr. Alok Mishra for his support.

REFERENCES

[1] Advanced card System Ltd. http://www.acs.com/
[2] Deepak Nagawade, *Implementation of SCOSTA-CL based Smart Card Operating System (SCSOS),* Indian Institute of Technology, Kanpur, India June 2008.
[3] IBM Deutschland Entwicklung GmbH Information Development, *MultiFunction Card 4.1 Family Programmer's Reference Version 1.0,* Dept. 3248 Postfach 1380 71003 Boeblingen Germany.
[4] IM Y. Jung, Sung I. Jun, Kyo I. Chung, *A Persistent Memory Management in Java Card*.
[5] Mastercard home page. http://www.mastercard.com/index.html/.
[6] PrEN 1545-4. *Identification Card Systems – Surface Transport Applications – Part 4*: Vehicle and Driver Licencing, 1997.
[7] R. Shankesi, *Developement of an Operating System for Smart Card. Master's thesis*, IndianInstitute of Technology Kanpur, Dept. of CSE, India, April 2002.
[8] Simple operating system for smart card education. http://www.mbsks.franken.de/sosse/.
[9] Visa International, USA home page. http://www.visa.com/.
[10] Visa, *Visa Smart Debit/Credit Certification Authority Service Description (VSDC) version 2.1*, December 2005.
[11] Wolfgang Rankl et al; *Smart Card Handbook Third Edition*, Giesecke and Devrient GmbH, Munich, Germany, John Wiley & Sons, Ltd publication.