

TFS4 Fast Unlink

Technical Paper

November-2007, Version 1.0





Copyright Notice

Copyright © 2007, Flash Software Group, Samsung Electronics Co., Ltd

All rights reserved.

Trademarks

TFS4 is a trademark of Memory Division, Samsung Electronics Co., Ltd in Korea and other countries.

Restrictions on Use and Transfer

No part of this guide may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Samsung Electronics Co., Ltd. as governed by Republic of Korea and international copyright laws.

Document confidentiality status: Restricted Distribution

Readers of this document are not granted the right to distribute this document to any third party without prior written agreement from Samsung Electronics Co., Ltd.

Samsung Electronics Co., Ltd may make changes to the contents of this document any time without notice.

THIS DOCUMENT IS INTENDED ONLY TO ASSIST THE READER IN THE USE OF THE PRODUCT. SAMSUNG ELECTRONICS Co., Ltd. SHALL NOT BE LIABLE FOR ANY LOSS OR DAMAGE ARISING FROM THE USE OF ANY INFORMATION IN THIS DOCUMENT. SAMSUNG ELECTRONICS Co., Ltd. ASSUMES NO RESPONSIBILITY FOR ANY ERROR OR OMISSION IN INFORMATION CONTAINED IN THIS DOCUMENT, OR ANY INCORRECT USE OF THE PRODUCT.

Contact Information

Flash Software Group
Memory Division
Samsung Electronics Co., Ltd

Address: San #16 Banwol-Dong, Taeon-Eup
Hwasung-City, Gyeonggi-Do, Korea, 445-701

Abstract

This Paper explains the Samsung's TFS4 FAT file system operation for effectively deleting a file without accessing the whole FAT entries on the FAT table.

Introduction

Samsung's TFS4 FAT file system uses a file allocation table (FAT) to manage the allocation of clusters on a disk. This FAT table is stored in a beginning portion of a disk separately from an area in which contents of files are stored. Each entry in a FAT for a disk is associated with a cluster of the disk. A file in a FAT file system having N clusters will use N entries in the FAT. Each cluster of a disk is assigned a unique address, called a cluster number. Cluster numbers are assigned consecutively, starting from 0. The cluster number for each cluster on the disk is used as an index into the FAT to access the FAT entry for that particular cluster on the disk. The directory entry of a FAT file system identifies the cluster number of the first cluster of a file. And the FAT entry that corresponds to the cluster number of the first cluster of a file identifies the cluster number of the second cluster of the file, and so on. In this fashion, the FAT entries corresponding to the cluster numbers on disk for the clusters of a file, form a linked list. The FAT entry for the last cluster of a file will contain value "0xffff", to indicate that this cluster is the last cluster of the file.

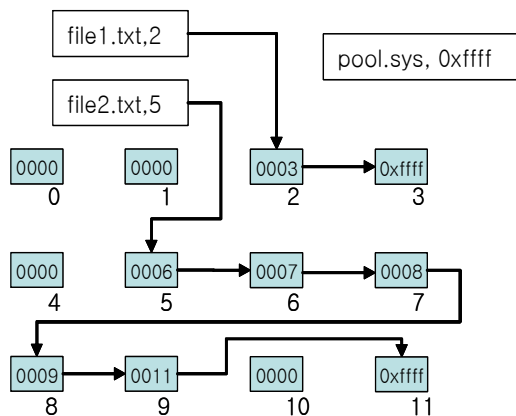


Figure 1. Linked List FAT Structure

As shown in FIG. 1, entry 4 has value 0 which indicates free cluster or an unallocated cluster. The value 0 tells the operating system which clusters are available for file assignments that require additional space. file1.txt has an initial cluster number of 2. This indicates that a first cluster of data for file1.txt

is stored in cluster 2 on the disk. This initial cluster 2 links to cluster 3 which is the last cluster containing data for this file. The file2.txt is also represented in a similar manner that has an initial cluster 5 and subsequent clusters 6,7,8,9,10 and the last cluster 11.

In conventional FAT file system, in order to delete a file, FAT entries are scanned one by one and updated to zero. If the FAT entry is long, this will take a long time. For example, consider a system where the FAT type is FAT16 and a file size is 40.96 MB, and if each cluster of disk consists of 8 sectors, then the total number of clusters to be allocated to the file system will be 10,000 and so 10,000 FAT entries will have to be updated in order to delete the file, which is a huge time consuming operation.

Fast Unlink methodology

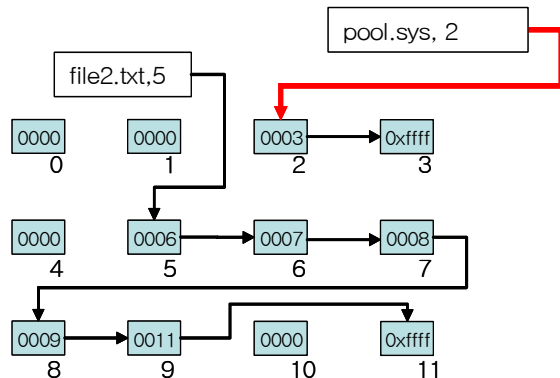


Figure 2. Unlink file1.txt

In order to solve this problem, Samsung's state of the art, fast unlink operation can quickly and efficiently reduce the time without accessing the whole FAT entries in the FAT table.

In the fast unlink operation, TFS4 initially creates a garbage file - pool.sys on the volume. This file is created automatically when the system is mounted. The pool.sys file is responsible for maintaining a pool of unlinked clusters and acts like a 'Recycle Bin'. Consider an example to delete file1.txt. First, the pool.sys file is linked to the initial cluster of file1.txt i.e. cluster 2 which links to cluster 3. Thus all the clusters belonging to file1.txt is linked to the pool.sys file. In the similar way, in order to delete file2.txt using fast unlink operation, the last cluster 3 of pool.sys file in FIG 2 will link to initial cluster of file2.txt i.e cluster 5 as shown in FIG 3.

Thus, the last cluster corresponding to pool.sys file will link to the initial cluster of a file to be deleted, during the fast unlink operation. It can be seen that with fast unlink, the FAT table need not be scanned sequentially to put zero in the FAT entry to indicate free cluster. Instead the files to be deleted are linked to the pool.sys file that acts like a 'Recycle Bin'. Thus it reduces the time to delete a file. Note that, even though the file is unlinked from the FAT entry, it does not result in contents of file actually being removed. To free the cluster space, delete the pool.sys file.

clusters are assigned sequentially from the pool.sys file. In FIG 4, the initial cluster 2 and subsequent clusters 3 and 5 from pool.sys file are allocated to newfile.txt. Thus the pool.sys file starts with initial cluster 6 and contains subsequent clusters 6, 7, 8, 9 and 11 ready to be allocated for a new file.

Conclusion

Samsung's Fast Unlink operation improves the file system performance by reducing the time for file delete operation and efficiently manages the cluster allocation for a new file.

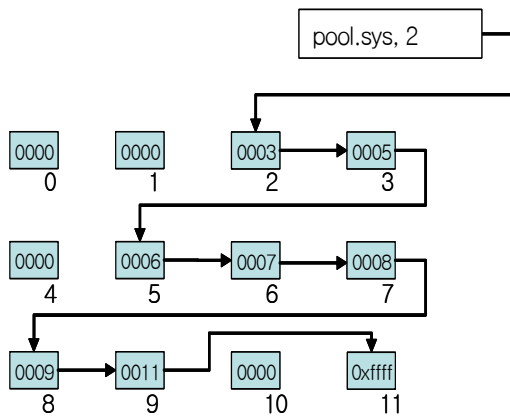


Figure 3. Unlink file2.txt

The pool.sys file also acts as a repository of space where all the unlinked files are linked together. So if there is a requirement of space, the whole FAT table need not be scanned to get free FAT entries, thus reducing the time. Instead, the pool.sys file is scanned to allocate unlinked clusters to a new file.

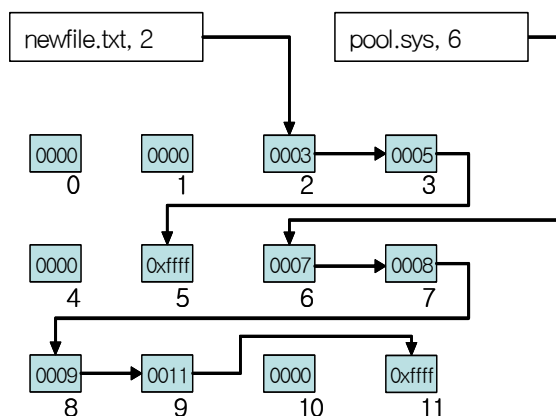


Figure 4. Cluster allocation from pool.sys file

As shown in FIG 4, in order to restore the unlinked clusters to a new file-newfile.txt that requires 3 cluster allocation, the initial cluster of pool.sys file is assigned to a new file and the subsequent three