CSC/CEG 3150 Assignment 3
File Recovery in FAT32

Background

In this assignment, you have to implement a C program that recovers a deleted file in FAT32. The suggested platform is Fedora Core 6 (FC6). To create a FAT32 partition in FC6 for this assignment, you should use a RAM-Disk. A RAM-Disk simulates a disk drive so that you don't have to create a real file FAT32 partition on your hard disk. The content in the RAM-Disk disappears when the computer is rebooted.

Two steps to use a FAT32 RAM-Disk:

1. Format a RAM-Disk as FAT32 format (e.g. /dev/ram)

   /sbin/mkfs.vfat -F 32 /dev/ram

2. Mount the RAM-Disk to a mount point (e.g. /mnt/rd)

   mount -t vfat /dev/ram /mnt/rd -o loop

It is recommended to write a shell script file to execute the above two commands as you may have to execute them many times throughout this assignment. You may use only one RAM-Disk in this assignment, so you can hard code the file system name (i.e. */dev/ram*) into your program instead of reading it from the input argument.

After you have created and mounted a RAM-Disk, you can see its status by using the *df* command.

The RAM-Disk can be operated by normal file operation commands on */mnt/rd*. For example, if you copy a file, say *abc.txt*, into */mnt/rd*, the result is copying the *abc.txt* file to the root directory of the RAM-Disk. Of course, you can freely create files or directories in */mnt/rd*.

You can unmount the RAM-Disk by the following command:

umount /mnt/rd

Milestone 1 - Print the information about the file system (10%)

In this part, you need to print the following information about the FAT32 file system when your program accepts an input argument *-i*.

1. Number of FATs
2. Number of bytes per sector
3. Number of sectors per cluster
4. Number of reserved sectors

The above information varies if different arguments are provided for the *mkfs.vfat* command when you are formatting the FAT32 file system. E.g.

/sbin/mkfs.vfat -F 32 -f 2 -S 512 -s 1 -R 32 /dev/ram

For more details, please refer to the man page of *mkfs.vfat* or *mkdosfs*. You cannot assume any sizes about the cluster size, the number of FATs, etc. You have to read the values from the boot sector.

Sample input and output

Assume that the RAM-Disk */dev/ram* is formatted as FAT32 by the above command and *Main* is the executable of your program.

```
[root]# ./Main -i
```

```
Number of FATs = 2
Number of bytes per sector = 512
Number of sectors per cluster = 1
Number of reserved sectors = 32
[root]#
```

Milestone 2 - List out the files and directories in the root directory (20%)

In this part, you need to list out all the file and directory in the root directory of the FAT32 file system when your program accepts an input argument *-l*. The output format is stated as follows:

1. You have to distinguish between files and directories by adding a '+' character at the end of a file name and a '/' character at the end of a directory name.

2. In the output, each line should list only one file/directory. If there are 3 files/directories in the root directory, then there are 3 lines in your output.

3. The listing order is not important.

Sample input and output

Assume that the root directory of */dev/ram* contains two files (*abc.txt* and *test1.txt*) and one directory (*dir1*).

```
[root]# ./Main -l
abc.txt+
dir1/
test1.txt+
[root]#
```

Here are some assumptions:

1. The name of a file has at most 8 characters and there must be an extension with exactly 3 characters. The file name and the extension only consists of a-z / 0-9.

2. The name of a directory has at most 8 characters. The directory name only consists of a-z / 0-9.

Milestone 3 - Recover a deleted file in the root directory (30%)

In this part, you need to recover a deleted file in the root directory of the FAT32 file system. You have to restore the directory entry and the content of the deleted file. For example, if *test1.txt* in the root directory is deleted by the *rm* command. It should be recovered by your program when your program accepts an input argument *-r* with the deleted file name, i.e. *-r test1.txt*.

Here are some assumptions:

1. The size of the deleted file is not greater than the size of a cluster of the file system.
2. The file name supplied by the user must be valid, i.e. it must refer to a deleted file with a correct name.
3. The name of the deleted file has at most 8 characters and there must be an extension with exactly 3 characters. The file name and the extension only consists of a-z / 0-9.
4. No file/directory creating or modifying (but not limited to file recovering) between the file deletion and the file recovery operations.

Sample input and output

Assume that the RAM-Disk */dev/ram* is formatted as FAT32 and it is mounted to */mnt/rd*.

```
[root]# cat /mnt/rd/test1.txt
This is test1.txt.
[root]# rm -f /mnt/rd/test1.txt
[root]# cat /mnt/rd/test1.txt
cat:  /mnt/rd/test1.txt:  No such file or directory
[root]# sync
[root]# ./Main -r test1.txt
[root]# umount /dev/ram
[root]# mount -t vfat /dev/ram /mnt/rd -o loop
[root]# cat /mnt/rd/test1.txt
This is test1.txt.
[root]#
```

When your program accepts an argument "*-r <filename>*", it recovers the specified file in the root directory of */dev/ram*. For the reason of having *sync*, *umount*, and *mount* commands before/after the execution of your program, please refer to the **Suggestions** section.

Milestone 4 - Recover a deleted file in a subdirectory (40%)

In this part, you need to recover a deleted file in a subdirectory of the FAT32 file system. You have to restore the directory entry and the content of the deleted file. For example, if */dir1/dir2/test2.txt* is deleted by the *rm* command. It should be recovered by your program when your program accepts an input argument *-d* with the deleted file name with *absolute path* corresponds to the FAT32 file system, i.e. *-d /dir1/dir2/test2.txt*.

Here are some assumptions:

1. The size of the deleted file is not greater than the size of a cluster of the file system.
2. The file name and the absolute path supplied by the user must be valid, i.e. they must refer to a deleted file with a correct absolute path and name.
3. The name of the deleted file has at most 8 characters and there must be an extension with exactly 3 characters. The file name and the extension only consists of a-z / 0-9.
4. The name of a directory has at most 8 characters. The directory name only consists of a-z / 0-9.
5. No file/directory creating or modifying (but not limited to file recovering) between the file deletion and the file recovery operations.
6. Only files are deleted, not directories.

Sample input and output

Assume that the RAM-Disk */dev/ram* is formatted as FAT32 and it is mounted to */mnt/rd*.

```
[root]# cat /mnt/rd/dir1/dir2/test2.txt
This is test2.txt.
[root]# rm -f /mnt/rd/dir1/dir2/test2.txt
[root]# cat /mnt/rd/dir1/dir2/test2.txt
cat:  /mnt/rd/dir1/dir2/test2.txt:  No such file or directory
[root]# sync
[root]# ./Main -d /dir1/dir2/test2.txt
[root]# umount /dev/ram
[root]# mount -t vfat /dev/ram /mnt/rd -o loop
[root]# cat /mnt/rd/dir1/dir2/test2.txt
This is test2.txt.
[root]#
```

When your program accepts an argument "*-d <filename with absolute path>*", it recovers the specified file in the specified subdirectory of */dev/ram*. Note that the absolute path corresponds to */dev/ram*, but not FC6. That means it is */dir1/dir2/test2.txt* instead of */mnt/rd/dir1/dir2/test2.txt*.

Suggestions

After you have made some changes to the file system, such as creating a file or deleting a file, you may find that the file system read by your program is not updated. It is because the kernel keeps data in memory instead of updating the data on disk in order to avoid doing relatively slow disk reads and writes. To solve this problem, you can use the *sync* command, which writes any data buffered in memory out to disk. You can either use the *sync* command before the execution of your program or call sync() inside your program.

Even if your program is correct, after it has recovered a deleted file, sometimes you still cannot find the file. To solve this problem, you can remount the RAM-Disk, i.e. unmount and then mount again (you may want to write a script file to do these), then you can find the recovered file.

Submission Guideline

For the submission of the assignment, please refer to the following link:

http://www.cse.cuhk.edu.hk/~csc3150/submission/

Due date: 23:59, December 2, 2007.

Bonus Part - To be released ...