**CS551 Operating Systems Design and Implementation**

**Project 3: Undelete**

**Design Document**

Group number: Fei Shen A20354692

Guanxin Wang A20356484

**1. UNDELETE**

**1.1 Manual page**

NAME

undelete -- recover any regular file that has been deleted in the current directory

SYNOPSIS

undelete file\_name

DESCRIPTION

undelete attempts to recover any regular file that has been deleted by the unlink() system call in the current directory. Alternatively, it will recover a specific file given its filename.

**1.2 Design of this tool**

To achieve the function of undelete, there are two parts: delete and recover. For the delete part, firstly, we should record the inodes information of the file which should be deleted and the parent directory. Secondly, we need to move the file to a temporary location. In our project, we design a customized struct to save parent\_inode, file\_inode, absolute path and file\_name as a table array stored in MFS servers. For the recovery part, firstly, we need to send parent directory inode info and target file name to MFS, let it search the inode of the file that needs to be recovered. Make sure the inode is not been used now (rip->i\_link == 0 or rip->i\_count == 0). If success, MFS send the target inode number to testing process. Then, we should search the file name in a temporary location with parent inode and file inode to match the former file. In the end, we move the file to the former location before it was deleted. In our project, we use the inode of the file to search the data in the table talked above.

**1.3 Possible exceptions and handling methods**

The possible exceptions and handling methods of undelete command are shown as table 1.

|  |  |
| --- | --- |
| **Table 1. UNDELETE POSSIBLE EXCEPTIONS** | |
| **possible exceptions** | **handing methods** |
| there was a file in the target location which has the same name as the name of the file needs to be undeleted | show message to the user ”the inode is being used , Recovery failed” |
| Rcmkdir never make, undelete the file needs to be deleted | show message to the user “no rcdirs” |
| Rcmkdir makes, no deleted files | show message to the user “no deleted files” |
| Found target inode number, but failed to get inode | Show message to the user “iNode is not there, Recovery failed” |
| when recover deleted file to its former location, there is a file with the same name | show message to the user “the inode is used” |
| Delete file not in recovery directory | Regular deletion without recording inodes of parent directory and file inodes, file name |

**2. RCMKDIR**

**2.1 Manual page**

NAME

rcmkdir -- make directories that undelete can only be used in these directories

SYNOPSIS

rcmkdir directory\_name

DESCRIPTION

rcmkdir is like the function of mkdir. The difference between rcmkdir and mkdir is that the users can undelete only on directories created by rcmkdir command.

**2.2 Design of this tool**

The function of rcmkdir is almost the same as the mkdir. The mainly difference is that the users can undelete only on directories created by rcmkdir command. To achieve this, when we undelete a file, we should check the path of the file by using the parent inode to make sure the file is from a directory made by rcmkdir command. Then, the undelete can become successful. To do so, we need to record every directory we created by rcmkdir command before. And when undelete files, we check the path with the records to find whether the directory is right to recover the files or not.

**2.3 Possible exceptions and handling methods**

The possible exceptions and handling methods of rcmkdir command are shown as table 2.

|  |  |
| --- | --- |
| **Table 2. RCMKDIR POSSIBLE EXCEPTIONS** | |
| **possible exceptions** | **handing methods** |
| attempt to undelete files from the directories not created by rcmkdir command | show message to the user “cannot undelete this file” |
| attempt to create a directory which have the same name as an existed one | show message to the user “cannot create this directory” |

**3. Bonus**

*If your design will allow the undeletable files to last “as long as possible”. Discuss what does “as long as possible” mean.*

“as long as possible” means that you can undelete the file after a very long time period, such as one week or longer. To allow this come true, we need to save the copy of the file in a safe location and keep the inode of the file unused.

at this point, we just send the file creation date as timestamp parameter to MFS servers while deleting and then we can save all deleted files with same name, therefore we just use LIFO algorithm to recover undeletablefile.

**4. Conclusion**

In this project, we need to have a deep understanding about MINIX file system structure and familiar about the functions such as unlink(), mv(), rm(), mkdir(), get\_inode(), put\_inode() and so on.

To achieve the project goal, we searched plenty of methods and documents about MINIX file system file recovery. It was harder to accomplish this project, but more knowledge we had learned about the file system.